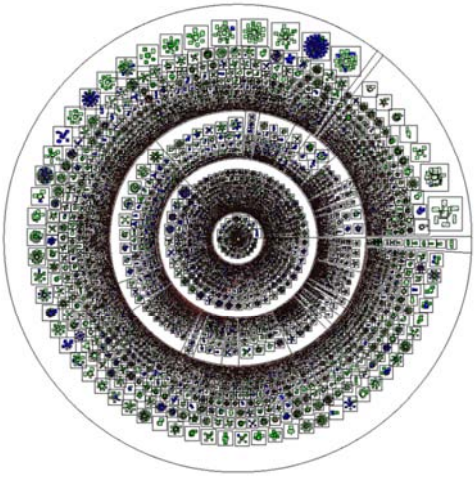
ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΔΙΑΤΜΗΜΑΤΙΚΟ ΠΡΟΓΡΑΜΜΑ
ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
«ΠΛΗΡΟΦΟΡΙΚΗ ΚΑΙ ΥΠΟΛΟΓΙΣΤΙΚΗ ΒΙΟΙΑΤΡΙΚΗ»

UNIX – LINUX: Kernels, distributions & security



Μάντζιος Παναγιώτης

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Υπεύθυνος
Λάμψας Πέτρος

Λαμία, 2016

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΔΙΑΤΜΗΜΑΤΙΚΟ ΜΕΤΑΠΤΥΧΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΠΛΗΡΟΦΟΡΙΚΗ ΚΑΙ ΥΠΟΛΟΓΙΣΤΙΚΗ ΒΙΟΙΑΤΡΙΚΗ

ΚΑΤΕΥΘΥΝΣΗ:

«ΠΛΗΡΟΦΟΡΙΚΗ ΜΕ ΕΦΑΡΜΟΓΕΣ ΣΤΗΝ ΑΣΦΑΛΕΙΑ, ΔΙΑΧΕΙΡΙΣΗ ΜΕΓΑΛΟΥ ΟΓΚΟΥ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΠΡΟΣΟΜΟΙΩΣΗ»

UNIX – LINUX: Kernels, distributions & security

Μάντζιος Παναγιώτης

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Επιβλέπων
Λάμψας Πέτρος

Λαμία, 2016

«Υπεύθυνη Δήλωση μη λογοκλοπής και ανάληψης προσωπικής ευθύνης»

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, και γνωρίζοντας τις συνέπειες της λογοκλοπής, δηλώνω υπεύθυνα και ενυπογράφως ότι η παρούσα εργασία με τίτλο «UNIX – LINUX: Kernels, distributions & security» αποτελεί προϊόν αυστηρά προσωπικής εργασίας και όλες οι πηγές από τις οποίες χρησιμοποίησα δεδομένα, ιδέες, φράσεις, προτάσεις ή λέξεις, είτε επακριβώς (όπως υπάρχουν στο πρωτότυπο ή μεταφρασμένες) είτε με παράφραση, έχουν δηλωθεί κατάλληλα και ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή. Αναλαμβάνω πλήρως, ατομικά και προσωπικά, όλες τις νομικές και διοικητικές συνέπειες που δύναται να προκύψουν στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής.

Ο ΔΗΛΩΝ
Μάντζιος Παναγιώτης

Ημερομηνία
18/07/2016

Υπογραφή

UNIX – LINUX: Kernels, distributions & security

Μάντζιος Παναγιώτης

Τριμελής Επιτροπή:

Λάμψας Πέτρος (επιβλέπων)

Λουκόπουλος Θανάσης

Αντωνής Κωνσταντίνος

“UNIX is simple. It just takes a genius to understand its simplicity.”

Dennis Ritchie

“The Linux philosophy is 'Laugh in the face of danger'. Oops. Wrong One. 'Do it yourself'. Yes, that's it.”

Linus Torvalds

Περίληψη

Τα λειτουργικά συστήματα του Linux και Unix χρησιμοποιούνται ευρέως από ένα μεγάλο αριθμό χρηστών, επιχειρήσεων αλλά και Server ανά τον κόσμο. Όμως πώς προέκυψαν τα συστήματα αυτά έτσι όπως τα γνωρίζουμε σήμερα; Τελικά τι είναι ένα λειτουργικό σύστημα; Είναι ο πυρήνας; Είναι τα προγράμματα που έχουμε εγκαταστήσει; Είναι οι διάφορες συνιστώσες και διεργασίες; Είναι το γραφικό περιβάλλον ή μήπως η γραμμή εντολών; Θα λέγαμε ότι είναι όλα τα παραπάνω και πολύ περισσότερα. Είναι κάθε στοιχείο που είναι εγκατεστημένο και αλληλεπιδρά με τον πυρήνα. Είναι κάθε στοιχείο που μπορούμε να δούμε αλλά και κάθε στοιχείο το οποίο εκτελείται στο προσκήνιο. Είναι εκείνο που είναι υπεύθυνο για όλες τις εργασίες που του ανατίθενται και συνδυάζει πόρους, υλικό, εφαρμογές με σκοπό την ευκολία της διεξαγωγής διαφόρων εργασιών.

Μία από τις πιο σημαντικές και καθοριστικές στιγμές στην ιστορία για την υλοποίηση των λειτουργικών συστημάτων είναι εκείνη της δημιουργίας της γλώσσας C. Αναπτύχθηκε στα εργαστήρια της Bell labs και αργότερα πάνω σε αυτή βασίστηκε το λειτουργικό σύστημα Unix. Από εκεί και έπειτα έχουμε μία αλματώδη ανάπτυξη των λειτουργικών συστημάτων, τα οποία ουσιαστικά αποτελούνται από πολλές επιμέρους συνιστώσες έτσι ώστε να βοηθούν τον χρήστη στις εργασίες που θέλει να υλοποιήσει.

Φτάσαμε όμως στο σημείο όπου υπάρχουν τόσες πολλές εφαρμογές, τόσες πολλές διεργασίες, τόσοι πολλοί παράγοντες που επηρεάζουν την ασφάλεια των λειτουργικών συστημάτων. Μπορούμε πράγματι να είμαστε ασφαλείς κάτω από την ομπρέλα του λειτουργικού μας συστήματος; Και αν όχι ποιες ενέργειες πρέπει να κάνουμε έτσι ώστε να έχουμε το επίπεδο ασφαλείας που χρειαζόμαστε; Υπάρχουν εργαλεία που μας παρέχουν προστασία; Ποιες εκδόσεις Linux και Unix είναι περισσότερο ασφαλείς και ποιες μπορούμε να παραμετροποιήσουμε; Μπορούμε να ελέγξουμε τον πυρήνα του Unix και του Linux έτσι ώστε να μας προστατεύει από ανεπιθύμητους εισβολείς; Σε όλα αυτά τα καίρια ερωτήματα και σε πολύ περισσότερα, καλείται να απαντήσει η παρούσα διπλωματική εργασία μέσα από μία διεξοδική έρευνα και αναζήτηση πληροφοριών.

Αναλυτικότερα στην παρούσα διπλωματική εργασία γίνονται αναφορές στα διάφορα στοιχεία και λειτουργίες των συστημάτων καθώς και στην υποδομή των πυρήνων που χρησιμοποιούν. Αναφερόμαστε στα στοιχεία που απαρτίζουν τα λειτουργικά συστήματα και δίνονται λεπτομέρειες σχετικά με τις διάφορες διανομές που υπάρχουν, ενώ παρουσιάζουμε διάφορες γνωστές ευπάθειες, όπως και τεχνικές αντιμετώπισης των διαφόρων κενών ασφαλείας.

Συγκεκριμένα στο πρώτο κεφάλαιο παρατίθενται λεπτομέρειες σχετικά με την εξελικτική πορεία των λειτουργικών συστημάτων, την γλώσσα προγραμματισμού C η οποία αποτέλεσε και αποτελεί τον ακρογωνιαίο λίθο υλοποίησης των λειτουργικών συστημάτων καθώς και την δομή αυτών.

Στο δεύτερο κεφάλαιο εμβαθύνουμε στο σύστημα του Unix και των λειτουργιών του πυρήνα αλλά και στις διάφορες διανομές που υπήρξαν και υπάρχουν. Ακόμη γίνεται σύγκριση των δύο λειτουργικών συστημάτων (Linux – Unix) και παρουσιάζονται οι βασικές τους διαφορές.

Στο τρίτο κεφάλαιο ως επίκεντρο είναι το Linux, οι λόγοι δημιουργίας του και οι διάφορες διανομές. Επίσης γίνεται ανάλυση των διαφόρων ευπαθειών και των μέτρων προστασίας.

Στο τέταρτο και τελευταίο κεφάλαιο εγκαθιστούμε την δημοφιλή διανομή slackware linux με σκοπό την παραμετροποίησή της, έτσι ώστε να μας παρέχει ένα συγκεκριμένο επίπεδο ασφάλειας. Πραγματοποιούμε ελέγχους με συγκεκριμένα εργαλεία ανίχνευσης εισβολέων, εν προκειμένω να ανακαλύψουμε τυχόν κενά ασφαλείας, ανεπαρκείς ρυθμίσεις προστασίας του συστήματος και τυχόν εισβολείς που έχουν καταφέρει να εισχωρήσουν στο σύστημά μας. Με βάση τα αποτελέσματα των εργαλείων ανίχνευσης γίνονται ενέργειες για την διασφάλιση της προστασίας του συστήματος από ανεπιθύμητους εισβολείς. Τέλος γίνεται σύντομη αναφορά στην νέα διανομή Linux tales που παρουσιάζει ιδιαίτερο ενδιαφέρον σε θέματα ασφαλείας και ανωνυμίας.

Abstract

The operating systems of Linux and Unix are widely used by a large number of users, enterprises and Servers all over the world. Many questions are raised such as how these systems are emerged as we know them today? Eventually, what is an operating system? Is it the core? Are the programmes that have been installed? Are the various components and processes? Is the GUI or the command line? We would say that it is all these things and much more. It is each component installed that interacts with the kernel. Every element that we see and every element running in the foreground. It is the element which is responsible for all tasks assigned and it is combining resources, hardware, applications in order to carry out various tasks.

One of the most important and defining moments in history for the implementation of operating systems is the creation of programming language C. This language is developed in Bell's labs and later on it was the basis for Unix operating system. From later on there is a rapid development of operating systems, which essentially consists of many individual components so as to assist the user in the work he wants to implement.

We have already reached the point where there are so many applications, so many processes, so many factors that affect the security of operating systems. Can we indeed be safe under the umbrella of our operating system? And if not what actions should we do so that we have the level of security we need? There are tools that provide us protection? What versions of Linux and Unix are more secure and which can we customize? Can we control the core of Unix and Linux so that it protects us from intruders? In all these critical questions and many more, this thesis is required to answer through an extensive research for further information.

Specifically, in this thesis there are references to various components and functions of the systems and infrastructure of cores used. We make references to elements that make up the operating systems, and provide details on various distributions available, while presenting several known vulnerabilities, as well as addressing the various security problem techniques.

Particularly, the first chapter gives details for the evolution of operating systems, the programming language C, which was and is the basis of the implementation of operational systems and their structure.

The second chapter is focusing on the Unix system and its core functions, but also the various distributions existed. There is also a comparison of the two operating systems (Linux - Unix) and a presentation of their main differences.

The third chapter is focusing on Linux, the reasons of creation and the various distributions. It is also attempted an analysis of various vulnerabilities and protective measures.

In the fourth and final chapter, we install the popular distribution slackware linux aiming its customization in order to give us a certain security level. We perform controls with specific intrusion detection tools, in order to find any flaws, an inadequate system protection settings and intruders that have managed to penetrate our system. Based on the results of detection tools, there are made actions to ensure the protection of the system from intruders. Finally, a brief reference is made for the new distribution Linux tales that presents a particular interest in security systems and anonymity.

Ευχαριστίες

Ευχαριστώ θερμά τον καθηγητή μου Δρ. Λάμψα Πέτρο, το Πανεπιστήμιο Θεσσαλίας που παρόλο που προέρχομαι από ΑΤΕΙ με αποδέχτηκε και μου έδωσε την ευκαιρία να εντρυφήσω περαιτέρω στο αντικείμενο της πληροφορικής. Όλους τους συναδέρφους που συνεργαστήκαμε και φέραμε εις πέρας όλα τα έργα που μας ανατέθηκαν. Το λοιπό προσωπικό και την γραμματεία για την πολύτιμη υποστήριξη στα διοικητικά θέματα. Τους φίλους και γονείς μου, που στέκονται δίπλα μου όλα αυτά τα χρόνια στο δύσκολο δρόμο τις εκπαιδευτικής διαδικασίας.

Πίνακας περιεχομένων

Κατάλογος σχημάτων

Κατάλογος Πινάκων

Εισαγωγή

Αδιαμφισβήτητα τα λειτουργικά συστήματα όπως είναι το Unix και το Linux κατέχουν μία ουσιώδη σημασία στον χώρο της πληροφορικής. Η δημιουργία τους αποτέλεσε και αποτελεί τομέα ερευνών για πολλούς επιστήμονες. Μεγάλοι υπερυπολογιστές αλλά και Server ανά τον κόσμο κάνουν χρήση των δύο αυτών λειτουργικών συστημάτων. Αυτό συμβαίνει διότι τα δύο αυτά συστήματα μπορούν να παραμετροποιηθούν ούτως ώστε να παρέχουν ασφάλεια και σταθερότητα, κάτι το οποίο είναι απαραίτητο σε συστήματα τα οποία διαχειρίζονται μεγάλο όγκο πληροφοριών και χρησιμοποιούνται ως κομβικά σημεία. Για να φτάσουμε όμως στο σημείο που είμαστε σήμερα και να χρησιμοποιούμε τα συστήματα αυτά μεσολάβησαν αρκετές δεκαετίες, οι οποίες μπορούν να χαρακτηριστούν ως εξελικτικές φάσεις των λειτουργικών συστημάτων γενικότερα. Από τα συστήματα που χρησιμοποιούσαν διάτρητες κάρτες μέχρι και το σήμερα, η εξέλιξη των συστημάτων (υλικό και λογισμικό) μπορεί επίσης να χαρακτηριστεί ως ραγδαία αναπτυσσόμενη. Αναπτυσσόμενη διότι συνεχώς αναπτύσσεται και δεν έχει σταματήσει, αλλά και ραγδαία διότι σε μόλις τέσσερις δεκαετίες έχουμε φτάσει στο σημείο να έχουμε στην τσέπη μας ένα λειτουργικό σύστημα με πολλές δυνατότητες.

Σε τακτά χρονικά διαστήματα βλέπουμε να αναπτύσσονται δεκάδες λειτουργικά συστήματα, όπως και διάφορες εκδόσεις αυτών. Τα περισσότερα βέβαια βασίζονται σε κάποιο ήδη ευρέως διαδεδομένο λειτουργικό σύστημα. Το Unix και το Linux κατέχουν θα λέγαμε στον τομέα αυτό την πρωτοκαθεδρία καθώς είναι η επιλογή πολλών κατασκευαστών για την δημιουργία άλλων λειτουργικών συστημάτων. Χαρακτηριστικά παραδείγματα αποτελούν το android που βασίζεται στο Linux αλλά και το Mac OS X που βασίζεται στο Unix. Εκατοντάδες εκδόσεις του Linux έχουν αναπτυχθεί μέχρι και σήμερα αλλά λίγες από αυτές κατέχουν μεγάλα μερίδια στην αγορά. Η χρήση του Linux ως βάση για την δημιουργία άλλων συστημάτων έγκειται στο γεγονός ότι το Linux είναι ένα ανοικτού τύπου λειτουργικό σύστημα και μπορεί να χρησιμοποιηθεί και να παραμετροποιηθεί για τις ανάγκες που το χρειάζεται η κάθε εταιρία χωρίς να πληρώσει δικαιώματα χρήσης σε εκείνους που το δημιούργησαν.

Κανείς δεν θα μπορούσε να φανταστεί κάποιες δεκαετίες πριν ότι θα φτάσουμε σε αυτό το σημείο, ούτε ότι η ασφάλεια των δεδομένων και των συστημάτων θα κατέχουν μία ουσιαστική θέση στην καθημερινότητα. Μεγάλες εταιρίες ανά τον κόσμο εξειδικεύονται σε θέματα ασφαλείας των συστημάτων γενικότερα, δημιουργώντας χιλιάδες θέσεις εργασίας στον τομέα αυτό. Σε καθημερινή βάση βλέπουμε να ανακαλύπτονται ευπάθειες σε κάθε λογής λογισμικό. Βλέπουμε επίσης τη ραγδαία εξάπλωση του κακόβουλου λογισμικού με άμεσες συνέπειες στην κλοπή ευαίσθητων πληροφοριών που σκοπό έχουν την αποκομιδή κερδών ή και άλλων δόλιων ενεργειών σε βάρος ανυποψίαστων θυμάτων ή και εταιρειών. Οι απειλές καθημερινά προσαρμόζονται στα νέα δεδομένα και στις νέες τεχνολογικές ανακαλύψεις κάνοντας ιδιαίτερα δύσκολο το έργο του εντοπισμού τους. Μαζί με αυτές φυσικά εξελίσσονται και τα διάφορα συστήματα, μέσα από την συνεχή ενημέρωση των διαφόρων συνιστωσών και εφαρμογών που τα αποτελούν. Για να κατανοήσουμε όλες τις εξελίξεις αλλά και το πως φτάσαμε μέχρι εδώ, είναι επιτακτική η ανάλυση και η διεξοδική έρευνα πάνω στα λειτουργικά συστήματα και τα μέρη που τα αποτελούν.

Κεφάλαιο 1

1.1 Πρώιμο στάδιο υλοποίησης των υπολογιστών και των λειτουργικών συστημάτων

Από το 1600 μ.Χ. θα μπορούσαμε να πούμε πώς άρχισε η σταδιακή ανάπτυξη της σκέψης αλλά και της δημιουργίας διαφόρων εφευρέσεων οι οποίες μπορούσαν να υπολογίζουν διάφορες αριθμητικές πράξεις. Στα επόμενα χρόνια αναπτύσσονται διάφορες κατασκευές όπως η πρώτη αθροιστική μηχανή που λειτουργούσε με οδοντωτούς τροχούς και μοχλούς, και μετέπειτα ο βελτιωμένος τύπος της μηχανής αυτής, που κατασκευάστηκε από τον Leibnitz το 1673, ο οποίος μπορούσε να εκτελεί και πολλαπλασιασμούς και διαιρέσεις. Μετέπειτα τον 19ο αιώνα, ο Charles Babbage (1792-1871) συνέλαβε την ιδέα της προγραμματισμένης εκτελέσεως των πράξεων και κατασκεύασε το 1822 την «μηχανή των διαφορών». Έντεκα χρόνια αργότερα, πάλι ο ίδιος επινόησε την «αναλυτική μηχανή» για την επεξεργασία μεγάλου όγκου αριθμητικών πληροφοριών. Όμως, η περιορισμένη τεχνολογία εκείνης της εποχής δεν επέτρεψε την πραγματοποίηση και ωφέλιμη χρήση αυτών των μηχανών. Τα επόμενα 100 χρόνια αναπτύσσονται περισσότερο η μαθηματική λογική, η θεωρία των αλγορίθμων, η θεωρία των αυτομάτων και άλλα.

Με την εγκαθίδρυση της βιομηχανικής κοινωνίας, η τεχνολογία φθάνει σε ένα βαθμό τελειότητας, απαραίτητο για την κατασκευή της πρώτης σύγχρονης υπολογιστικής μηχανής. Ο Howard Aiken με την βοήθεια της IBM παρουσιάζει το 1944 τον Mark I. Ο Mark I χρειάζεται 0,3sec για μία πρόσθεση ή αφαίρεση, 4sec για ένα πολλαπλασιασμό και 10sec για μία διαίρεση. Ένα έτος αργότερα, οι Eckert και Mauchly κατασκευάζουν τον ENIAC (Electronic Numerical Integrator And Calculator). Αυτός είναι ο πρώτος σύγχρονος Ηλεκτρονικός Υπολογιστής με ηλεκτρονικές λυχνίες, ρευματονόμους κ.λ.π., και είναι 100 φορές ταχύτερος από τον Mark I. Από τότε παρατηρείται ραγδαία ανάπτυξη των ΗΥ οι οποίοι αρχίζουν πλέον να μπαίνουν στη ζωή του κοινού ανθρώπου. Η ανάπτυξη αυτή χωρίζεται σε διάφορες « εποχές» ή «γενιές» ΗΥ. Η πρώτη γενιά έχει σαν βασικό χαρακτηριστικό της την χρήση της ηλεκτρονικής λυχνίας με αντιπροσώπους τον ENIAC και άλλους. Οι ΗΥ της δεύτερης γενιάς (περί το 1955) χρησιμοποιούν ημιαγωγούς (transistors) και ο χρόνος εκτέλεσης μίας πράξεως είναι της τάξεως των 100μsec . Στην τρίτη γενιά (μετά το 1965) έχουμε τα ολοκληρωμένα κυκλώματα. Ο χρόνος που απαιτείται για την εκτέλεση μίας πράξεως είναι της τάξεως του 1μsec. Χαρακτηριστικό της εξέλιξης αυτής είναι και η συνεχής μείωση του όγκου του ΗΥ. Αυτό είναι αποτέλεσμα της αντικατάστασης των λυχνιών από ημιαγωγούς και στη συνέχεια από ολοκληρωμένα κυκλώματα.

Περί τα τέλη της δεκαετίας του 1970 με αρχές του '80-' αρχίζουν και εμφανίζονται οι πρώτοι «προσωπικοί υπολογιστές» οι οποίοι διέθεταν ένα υποτυπώδες γραφικό περιβάλλον (GUI-Graphical User Interface) έτσι όπως το γνωρίζουμε σήμερα. Είναι σχετικά φθηνοί και καταλαμβάνουν μικρό όγκο. Η ραγδαία εξέλιξή τους με την παράλληλη μείωση της τιμής τους είχε σαν αποτέλεσμα να φθάσουν οι ΗΥ στο σημείο που είμαστε σήμερα. Ταυτόχρονα, η χρήση άλλων τεχνικών όπως η παράλληλη επεξεργασία, τα κατανεμημένα συστήματα και άλλες επιφέρουν την δραματική αύξηση των δυνατοτήτων των ΗΥ.

Τέλος, ένα σημαντικό βήμα στην ανάπτυξη των ΗΥ ήταν και η επινόηση και χρήση των δικτύων ΗΥ. Από τα τέλη της δεκαετίας του 1950, οι επιστήμονες είχαν αρχίσει να πειραματίζονται με την δικτύωση πολλών ΗΥ. Περί τα μέσα της επόμενης δεκαετίας, ήδη τα δίκτυα ΗΥ χρησιμοποιούνται από αεροπορικές εταιρείες, Τράπεζες, Στρατό και πολλές άλλες επιχειρήσεις. Το αποκορύφωμα αυτής της τεχνολογίας, οδήγησε στην γέννηση του Διαδικτύου (Internet) με τις γνωστές πλέον χρήσεις και δυνατότητές του. (**Κοκκινόπουλος, Σάββας**, 2008)

1.2 Η ανάπτυξη της γλώσσας C ως καθοριστικός παράγοντας στην υλοποίηση των λειτουργικών συστημάτων και ειδικότερα του UNIX-LINUX

Η συγγραφή των διαφόρων λειτουργικών συστημάτων όπως των MS Windows αλλά κυρίως του UNIX από το οποίο προέκυψαν το GNU/LINUX, τα Mac OS, το BSD, το Solaris και άλλα, βασίζεται πάνω στην ευρέως διαδεδομένη γλώσσα C που στην συνέχεια μετεξελίχθητε στην C++.

Σύμφωνα με τους **Kernighan & Ritchie** στο έργο τους **«C programming language»** η C είναι μια διαδικαστική (procedural)[1] γλώσσα προγραμματισμού γενικής χρήσης, η οποία αναπτύχθηκε αρχικά, μεταξύ του 1969 και του 1973, από τους ίδιους, στα εργαστήρια AT&T Bell Labs για να χρησιμοποιηθεί για την ανάπτυξη του λειτουργικού συστήματος UNIX. Όπως οι περισσότερες διαδικαστικές γλώσσες προγραμματισμού που ακολουθούν την παράδοση της ALGOL[2], η C έχει δυνατότητες δομημένου προγραμματισμού και επιτρέπει τη χρήση αναδρομής (αλλά όχι και εμφωλευμένων συναρτήσεων), ενώ, ο στατικός ορισμός του τύπου των μεταβλητών που επιβάλλει, προλαμβάνει πολλά σφάλματα κατά την χρήση τους. Ο σχεδιασμός της περιλαμβάνει δομές που μεταφράζονται αποδοτικά σε τυπικές εντολές μηχανής (machine instructions) και εξ αιτίας αυτού χρησιμοποιείται συχνά σε εφαρμογές που παλιότερα γράφονταν σε συμβολική γλώσσα (assembly language)[3]. Αυτό ακριβώς το χαρακτηριστικό της, έχει σαν συνέπεια και την αυξημένη ταχύτητα εκτέλεσης των εφαρμογών που γράφονται σε αυτή, καθώς και το γεγονός ότι είναι διαθέσιμη στα περισσότερα σημερινά λειτουργικά συστήματα, συνέβαλε κατά πολύ στην καθιέρωση της και την χρήση της για ανάπτυξη λειτουργικών συστημάτων και λοιπών προγραμμάτων συστήματος (system software), αλλά και απλών εφαρμογών.

Η C συγκαταλέγεται πλέον στις πιο ευρέως χρησιμοποιούμενες γλώσσες προγραμματισμού όλων των εποχών και πολλές νεώτερες γλώσσες έχουν επηρεαστεί άμεσα ή έμμεσα από αυτήν, συμπεριλαμβανομένων των C++[4], C#[4], D[5], Go[6], Java[7], JavaScript[7], Limbo[8], LPC[9], Perl[10], PHP[11], Python[12], καθώς και του κελύφους C (C shell[13]) του Unix. Κάποιες από αυτές τις γλώσσες έχουν επηρεαστεί κυρίως στη σύνταξή τους, με το σύστημα τύπων, τα μοντέλα δεδομένων και το νόημα των εκφράσεων τους να διαφέρουν σημαντικά από την C. Η C++, ειδικά, ξεκίνησε σαν προεπεξεργαστής της C, αλλά έχει εξελιχθεί πλέον σε μια αντικειμενοστραφή γλώσσα, που αποτελεί υπερσύνολο της C (**Wikipedia**, 2016)

Ειδικότερα όσον αφορά τα λειτουργικά συστήματα, τους πυρήνες (kernels) αλλά και διάφορα άλλα μέρη των συστημάτων η γλώσσα C (καθώς επίσης και σε αρκετά άλλα μέρη των συστημάτων η C++) είναι εκείνη που χρησιμοποιείται ευρέως

για την υλοποίηση τους. Αυτό συμβαίνει για διαφόρους λόγους μεταξύ των οποίων είναι:

1. Φορητότητα. Η C είναι γλώσσα που μοιάζει περισσότερο με τον κώδικα μηχανής, διατηρώντας παράλληλα τη δυνατότητα μεταφοράς μεταξύ των διαφόρων αρχιτεκτονικών.
2. Άμεση πρόσβαση στη μνήμη. Η διαχείριση της μνήμης αφήνεται στον προγραμματιστή ο οποίος πρέπει να χειριστεί καταλλήλως την κατανομή και ανακατανομή της μνήμης. Ως εκ τούτου αυτό αποτελεί πλεονέκτημα όσον αφορά την ανάπτυξη του λειτουργικού συστήματος (OS) διότι πρακτικώς όλα τα μέρη σε ένα σύστημα εμπλέκονται με τον άμεσο χειρισμό των περιοχών της μνήμης RAM.
3. Δεν υπάρχουν εξαρτήσεις χρόνου εκτέλεσης. Σε αντίθεση με την C ++, Java, C# και άλλες, η C δεν έχει καθόλου εξαρτήσεις χρόνου εκτέλεσης. Δεν αναμένει κανένα πρόγραμμα διαχείρισης μνήμης, δεν αναμένει τη διαχείριση των διαδικασιών, η μόνη runtime εξάρτηση που έχει είναι όταν καλεί τη συνάρτηση main()[14]. Αυτό σημαίνει ότι μπορούμε να γράψουμε ένα πρόγραμμα σε C, που τρέχει απευθείας από το υλικό, χωρίς να περιμένουμε κάποιον διαχειριστή μνήμης να εκτελεστεί.
4. Αποτελεσματική διακίνηση και διαχείριση σε επίπεδο byte. Η C είναι απολύτως κατάλληλη για την αποτελεσματική διαχείριση στο επίπεδο byte. Δηλαδή, την αντιγραφή δεδομένων σε διάφορους τομείς, κατανέμοντας κατάλληλα τα bits τριγύρω, και την επεξεργασία των δεδομένων σε μικρές ποσότητες. Ενώ αυτό είναι πιο κουραστικό για τον προγραμματιστή, είναι πολύ πιο αποδοτικό, τόσο στη χρήση της μνήμης RAM όσο και στο χρόνο επεξεργασίας, το οποίο είναι απαραίτητο σε λειτουργικά συστήματα τύπου T-kernel[15].
5. Όλος ο κώδικας της γλώσσας C έχει άμεση 1-1 μετάφραση, σε κώδικα μηχανής που είναι ευκόλως κατανοητή. Ως αποτέλεσμα, η C δεν έχει καμία build in πολύπλοκη δομή δεδομένων, όπως τα δέντρα (trees) ή τους πίνακες κατακερματισμού (hash). Η φιλοσοφία της C (και του UNIX) είναι ότι ο προγραμματιστής ξέρει καλύτερα τη γλώσσα, και αυτό την καθιστά ιδιαίτερα κατάλληλη για την τελειοποίηση των λεπτομερειών του λειτουργικού συστήματος (OS) και την ανάπτυξη των συστημάτων.

Εκτός από τους παραπάνω λόγους η C είναι εκείνη που χρησιμοποιήθηκε όπως είπαμε και προηγουμένως για την ανάπτυξη του λειτουργικού συστήματος UNIX (αναλύεται στο επόμενο κεφάλαιο) από το οποίο προέκυψαν αρκετά σύγχρονα λειτουργικά συστήματα και ειδικότερα τα Linux των οποίων οι βασικές αρχές στηρίζονται στο UNIX. Ο πυρήνας (Kernel) του Linux, και τα περισσότερα άλλα μέρη του λειτουργικού συστήματος, είναι επίσης γραμμένα (κυρίως) σε γλώσσα C. Το 70% του Linux kernel αναπτύχθηκε σε C, ένα 25% έχει αναπτυχθεί σε C ++ και το υπόλοιπο κατανέμεται σε διάφορες γλώσσες, όπως η Perl[10]. Ειδικότερα ο πυρήνας και οι οδηγοί για τα περιφερειακά: λειτουργούν σε ένα πραγματικά χαμηλό επίπεδο εργασιών στον υπολογιστή. Για να γράψουμε τον πυρήνα του λειτουργικού συστήματος και να αποκτήσουμε πρόσβαση στις ιδιότητες του υλικού, όπως τους κύκλους εισόδου του διαύλου μνήμης / εξόδου, κλπ, μια γλώσσα που μπορεί να επικοινωνήσει με την τροφοδοσία του υλικού είναι απαραίτητη. Οι βιβλιοθήκες

(Libraries) και οι υπηρεσιών κοινής ωφελείας (Utilities) και τα βασικά βοηθητικά προγράμματα του συστήματος, όπως τα: mkdir[16], chmod[16] , chown[16] , head[16] , tail[16] , chroot[16] , uptime[16], users είναι και αυτά γραμμένα στην C. Διαχειριστές πακέτων και λογισμικού διαμόρφωσης όπως το yum[17], apt[17], dpkg[17], κλπ, είναι επίσης γραμμένα σε C, καθώς και πληθώρα άλλων συνιστωσών όπως τα διάφορα περιβάλλοντα επιφάνειας εργασίας, οι διαχειριστές παραθύρων κλπ.

Εκτός του UNIX-LINUX και άλλα συστήματα και συνιστώσες αυτών βασίζουν την ανάπτυξή τους στην C. Πιο συγκεκριμένα ο πυρήνας των MS Windows αναπτύχθηκε σε C και τα υπόλοιπα μέρη του σε C++ και C#. Ο πυρήνας των Mac της εταιρίας Apple αναπτύχθηκε και αυτός στην γλώσσα C και στην αντικειμενοστραφή C. Τα BSD συστήματα βασίζονται και αυτά στο UNIX οπότε η βασική τους ανάπτυξη είναι η γλώσσα C.

---

[1]Το πρόγραμμα είναι οργανωμένο σε διαδικασίες, που αποτελούνται από σειρές εντολών που περιγράφουν αλγορίθμους

[2]History of ALGOL, Computer history museum, software preservation group

[3]Εγχειρίδιο Χρήσης AT91, Χαρίδημος Βέργος, Νικόλαος Κωστάρας, Πανεπιστήμιο Πατρών, 2007

[4]Η C++ περιλαμβάνει όλες εκείνες τις συνιστώσες της C, και επιπλέον προσθέτει την ενθυλάκωση, τον πολυμορφισμό, τα πρότυπα και μια ολόκληρη πρότυπη βιβλιοθήκη. Οι διαφορές μεταξύ των C και C ++ είναι λεπτές. Η C++ είναι απλά μια καθαρότερη έκδοση της C με ορισμένες βελτιώσεις και προσθήκες. Βλέπε The C++ programming language, Bjarne Stroustrup (creator of the c++), fourth edition, 2013 και Object-oriented Programming in C#, Kurt Normark, Aalborg University, Denmark 2010

[5]The D Programming Language, Andrei Alexandrescu, Pearson Education, 2010

[6]The Go Programming Language, Alan A. A. Donovan & Brian W. Kernighan, 2016

[7]Introduction to Programming Using Java, David J. Eck, Hobart and William Smith Colleges, 1996–2014 και JavaScript: The Definitive Guide, Sixth Edition, David Flanagan, O’Reilly Media, 2011

[8]Inferno Programming With Limbo, Phillip Stanley-Marbell Carnegie Mellon University, 2003

[9]Η LPC είναι μία αντικειμενοστραφείς γλώσσα προγραμματισμού που έχει επηρεαστεί από την C και την C++. Δημιουργήθηκε από τον Lars Pensjö

[10]Perl for Beginners, Geoffrey Sampson & Ventus Publishing ApS, 2010

[11]Programming PHP, Third Edition, Kevin Tatroe, Peter MacIntyre, and Rasmus Lerdorf, O’Reilly Media, 2013

[12]Learning to program with python, Richard L. Halterman, 2011

[13]Το κέλυφος C είναι ένα κέλυφος του Unix που δημιουργήθηκε από τον Bill Joy, ενώ ήταν ακόμη μεταπτυχιακός φοιτητής στο Πανεπιστήμιο της Καλιφόρνιας, Berkeley στα τέλη της δεκαετίας του 1970. Έχει διανεμηθεί ευρέως, ξεκινώντας με την κυκλοφορία 2BSD του συστήματος BSD Unix που ο Bill Joy άρχισε να διανέμει το 1978. Το κέλυφος C είναι ένας επεξεργαστής εντολών ο οποίος τρέχει σε ένα παράθυρο κειμένου, επιτρέποντας στο χρήστη να πληκτρολογεί εντολές

[14]Η εντολή main καλείται κατά την εκκίνηση του προγράμματος, μετά την εκκίνηση των μη-τοπικών αντικειμένων με στατική διάρκεια αποθήκευσης. Είναι το καθορισμένο σημείο εισόδου σε ένα πρόγραμμα που εκτελείται στο περιβάλλον που φιλοξενείται (δηλαδή, το λειτουργικό σύστημα).

[15]Ο T-Kernel είναι ένα ανοιχτού κώδικα λειτουργικό σύστημα σε πραγματικό χρόνο σχεδιασμένα για μικροελεγκτές 32-bit. Έχει τυποποιηθεί από το Φόρουμ T-Engine., η οποία το διανέμει βάσει της σύμβασης άδειας χρήσης "T-License".

[16]Κεφάλαιο 2^{ο}

[17]Κεφάλαιο 3^{ο}

1.2.1 Θέματα ασφαλείας στη γλώσσα C

Δεν είναι λίγες οι φορές που μεγάλα η μικρά θέματα ασφαλείας επιτρέπουν σε επίδοξους εισβολείς να υποβαθμίσουν την ασφάλεια των συστημάτων. Ειδικότερα στους τομείς της ανάπτυξης των λειτουργικών συστημάτων τα οποία αποτελούν τον ακρογωνιαίο λίθο πάνω στον οποίο στηρίζονται οι κατεξοχήν εφαρμογές και κάθε είδους εργασία που εκτελείτε στα συστήματα, το θέμα ασφάλειας θα πρέπει να αντιμετωπίζεται ως η κατεξοχήν νούμερο ένα προτεραιότητα των προγραμματιστών και των οίκων ανάπτυξης λογισμικού.

Η παράβλεψη των πρακτικών ασφαλείας μπορεί να έχει ολέθριες συνέπειες με αποτέλεσμα όταν ανακαλύπτονται κενά στον κώδικα συγγραφής, οι προγραμματιστές να καλούνται να τον διορθώσουν. Αυτό μπορεί να έχει ως αποτέλεσμα και την οικονομική επιβάρυνση των προγραμματιστών ή των οίκων ανάπτυξης καθώς απαιτείται η διακοπή των όποιων εργασιών και η ασχολία για την επιδιόρθωση των προβλημάτων. Γι αυτό η ανάπτυξη των συστημάτων θα πρέπει να διακατέχεται από την διασφάλιση της ποιότητας του λογισμικού. Επομένως οι προγραμματιστές θα πρέπει να έχουν γνώση όλων των πιθανών ευπαθειών ασφαλείας της γλώσσας και του περιβάλλοντος προγραμματισμού, ώστε αυτές να αποφεύγονται οι να αντιμετωπίζονται με κατάλληλους τρόπους και επίσης να γίνονται συχνές επιθεωρήσεις του κώδικα με σκοπό των εντοπισμό των πιθανών σημείων που μπορεί να διακυβεύεται η ασφάλεια.

Δεδομένου ότι η C είναι η πιο διαδεδομένη γλώσσα για την ανάπτυξη λογισμικού και επειδή οι περισσότερες γλώσσες κάνουν χρήση των βιβλιοθηκών της C, θα κάνουμε αναφορά μόνο στην γλώσσα αυτή, από την άποψη της ασφάλειας που στην παρούσα διπλωματική εργασία αποτελεί και μιζών θέμα. Παρόμοιες ευπάθειες όπως οι παρακάτω μπορούμε να βρούμε και σε άλλες γλώσσες προγραμματισμού.

Όπως αναφέρουν και οι καθηγητές **Nihal A. D'Cunha, Massimiliano Pala και Sean W. Smith** του Πανεπιστήμιου του **Dartmouth,** τα σφάλματα μνήμης είναι τα συνηθέστερα χρησιμοποιούμενα μέσα, με τα οποία οι εισβολείς προσπαθούν να αποκτήσουν τον έλεγχο μιας εφαρμογής η ενός συστήματος. Οι δικλείδες ασφαλείας της μνήμης συνεπώς, αποτελούν προϋπόθεση για την ασφάλεια, κατά των επιθέσεων που εκμεταλλεύονται τα κενά στον κώδικα. Οι επιτιθέμενοι εκμεταλλεύονται αυτά τα τρωτά σημεία, για τη μεταφορά του ελέγχου των προγραμμάτων σε κώδικα που έχουν εισάγει (το οποίο συνήθως είναι να ξεκινήσει ένα νέο κέλυφος (shell) με δικαιώματα διαχειριστή). Ένα πρόγραμμα το οποίο είναι ασφαλές από τρωτά σημεία στη μνήμη ποτέ δεν θα γράψει απρόσεκτα πάνω από τη μνήμη.

Όπως επίσης μας επισημαίνει κι ο καθηγητής **Κ. Βασιλάκης** του **Καποδιστριακού Πανεπιστημίου Αθηνών** κατά κύριο λόγω οι γλώσσες C και C++ παρουσιάζουν τις περισσότερες ευπάθειες σε σχέση με άλλες γλώσσες προγραμματισμού και αυτό συμβαίνει διότι είναι οι κύριες γλώσσες πάνω στις οποίες αναπτύσσονται λογισμικά εφαρμογών και συστημάτων, όπως είναι και τα λειτουργικά συστήματα, τα παραθυρικά περιβάλλοντα, οι βάσεις δεδομένων κλπ. Επίσης όπως αναφέρθηκε και σε προηγούμενη ενότητα οι γλώσσες αυτές έχουν σχεδιαστεί με γνώμονα ότι ο προγραμματιστής γνωρίζει καλά τι κάνει, επιτρέποντας λειτουργίες που σε άλλες γλώσσες θα απαγορεύονταν συνολικά. Παραδείγματα τέτοιον ενεργειών είναι: η αυθαίρετη μετατροπή δεικτών από έναν τύπο δεδομένων σε έναν άλλο, η κλήση συναρτήσεων με μεταβλητό αριθμό παραμέτρων, η χρήση αρνητικών δεικτών σε έναν πίνακα κ.ο.κ.

Η γλώσσα C είναι ευάλωτη σε επιθέσεις εγχύσεως κώδικα (injection code) (**Turner,** 2005). Κατανέμει τη μνήμη και τις τοπικές μεταβλητές σε λειτουργία (αυτόματη), καθολικές μεταβλητές (static) και malloc (δυναμική). Μερικοί προγραμματιστές δεν γνωρίζουν ότι είναι υπεύθυνοι για την κατανομή και την ανακατανομή της μνήμης. Μεταφραστές - που μετατρέπουν τον κώδικα σε ένα εκτελέσιμο πρόγραμμα - δεν ανιχνεύουν τα κοινά σφάλματα στη C κατά το χρόνο εκτέλεσης. Τα λάθη περιλαμβάνουν δείκτες για την ανακατανομή μνήμης και διατήρηση της μνήμης επ 'αόριστον αφού χρησιμοποιηθεί.

Πολλά επίσης από τα προβλήματα ασφαλείας σχετίζονται με την υπερχείλιση μνήμης. Όταν αναφερόμαστε στην υπερχείλιση μνήμης εννοούμε τις περιπτώσεις όπου δεσμεύεται ένα ποσό μνήμης για κάποια συγκεκριμένη λειτουργία, αλλά κατά την εκτέλεση της λειτουργίας απαιτείται μεγαλύτερο ποσό μνήμης από εκείνο που δεσμεύτηκε αρχικά. Δύο είναι οι αναγκαίες συνθήκες για να έχουμε υπερχείλιση μνήμης:

- ✓ Μη αξιόπιστες πηγές στη λήψη κάποιον δεδομένων (δίκτυο, αρχείο, πληκτρολόγιο, διαδιεργασιακή επικοινωνία κλπ)
- ✓ Αποθήκευση των δεδομένων σε ενδιάμεση μνήμη περιορισμένου μεγέθους χωρίς κατάλληλο έλεγχο αν τα δεδομένα όντως χωράνε στη μνήμη

Το αποτέλεσμα είναι να επικαλυφθούν τιμές από γειτονικές μεταβλητές ή/και διευθύνσεις επιστροφής από συναρτήσεις, τιμές αποθηκευμένων καταχωρητών κ.ο.κ. (**Βασιλάκης**, 2005)

1.2.1.1 Ασφαλής προγραμματισμός στη C

Διάφοροι τύποι ασφαλούς προγραμματισμού έχουν αναπτυχθεί. Οι ασφαλείς γλώσσες προγραμματισμού είναι γλώσσες στις οποίες το μεγαλύτερο μέρος των ανωτέρω ευπαθειών και πολλών άλλων, έχουν γίνει δύσκολο να γίνουν ή έχουν εξαλειφθεί πλήρως. Με την κωδικοποίηση σε αυτές τις ασφαλέστερες γλώσσες είναι απίθανο ότι τα προγράμματα θα υποφέρουν από κοινά τρωτά σημεία στην ασφάλεια. Για να επωφεληθούν από αυτές τις γλώσσες, οι προγραμματιστές πρέπει είτε να εφαρμόσουν σε ένα πρόγραμμα τη χρήση τους, είτε να εισάγουν ένα υπάρχον πρόγραμμα σε αυτές.

Οι παρακάτω τύποι ασφαλούς προγραμματισμού (**D'Cunha** et al) αναφέρονται στις περιπτώσεις τις γλώσσας C και είναι οι εξής:

- ✓ Η CCured είναι ένας ασφαλής τύπος εφαρμογής της C που στατικά επιχειρεί να βεβαιώσει ότι ο πηγαίος κώδικας είναι απαλλαγμένος από σφάλματα μνήμης, και εισάγει ελέγχους χρόνου εκτέλεσης, όπου η στατική ανάλυση δεν εγγυάται την ασφάλεια. Η CCured επιδιώκει να μετατρέψει τα προγράμματα γραμμένα σε C σε εκδοχές ισοδύναμες με την ασφάλεια της μνήμης, και ο κύριος στόχος της είναι να φέρει ασφάλεια στις εφαρμογές παλαιού τύπου. Μπορεί επίσης να λειτουργήσει και ως ένα εργαλείο εντοπισμού σφαλμάτων (debugging tool), δεδομένου ότι απαιτεί ένα πρόγραμμα να είναι ασφαλές από ευπάθειες στη μνήμη. Αποτελείται από διάφορα στοιχεία: έναν OCaml

μεταφραστή, ένα σύνολο απο Perl scripts που χρησιμοποιούνται για να επικαλεστούν την CCured εφαρμογή, και μια βιβλιοθήκη χρόνου εκτέλεσης.

- ✓ Η Cyclone είναι ένας τύπος ασφαλούς διαλέκτου της C, η οποία επιβάλλει ορισμένους περιορισμούς για τη διατήρηση της ασφάλειας, και προσθέτει κάποια χαρακτηριστικά για να ανακτήσει κοινά ιδιώματα προγραμματισμού με ασφαλή τρόπο. Η Cyclone διατηρεί την σύνταξη και τη σημασιολογία της C, ενώ προλαμβάνει τις πιο κοινές ευπάθειες ασφαλείας που υπάρχουν σε προγράμματα της C. Εμποδίζει τις παραβιάσεις ασφαλείας σε προγράμματα με ένα τρόπο παρόμοιο με εκείνο της CCured, δηλαδή με τη χρήση ενός συνδυασμού της στατικής ανάλυσης για τη διαχείριση της μνήμης και του ελέγχου του χρόνου εκτέλεσης για τα όρια παραβιάσεις. Η Cyclone εξασφαλίζει την ασφάλεια των τύπων, ενώ προσπαθεί να διατηρήσει των έλεγχο του χαμηλού επιπέδου πάνω από πτυχές όπως η διαχείριση μνήμης. Σύμφωνα με τους **D'Cunha** et al (όπως αναφέρεται στους **Grossman** et al, 2002), μία από τις πιο ενδιαφέρουσες πτυχές της Cyclone είναι οι εφαρμοζόμενοι μηχανισμοί πρόληψης για τον χειρισμό των διαφόρων τύπων δεικτών που δεν οδηγούν σε έγκυρα αντικείμενα και των διαρροών μνήμης.
- ✓ Η Vault, όπως η Cyclone, είναι μια ασφαλής έκδοση προγραμματισμού της C με ένα σύστημα μονάδας και ένα νέο χαρακτηριστικό για προσδιορισμό και έλεγχο των πόρων του προγράμματος (που αντιπροσωπεύεται από κλειδιά). Οι προγραμματιστές έχουν τη δυνατότητα να ελέγξουν τη διάταξη των δεδομένων και τη διάρκεια των πόρων των προγραμμάτων (όπως η μνήμη) ενώ παρέχεται με εγγυήσεις ασφάλειας.
- ✓ Η αντικειμενοστραφής Caml (OCaml) ανήκει στην Μετα-γλωσσική (ML) οικογένεια των γλωσσών προγραμματισμού. Είναι η πιο δημοφιλή και εκτεταμένα χρησιμοποιούμενη έκδοση της Caml – μία γενικού σκοπού γλώσσα προγραμματισμού, σχεδιασμένη για την ασφάλεια των προγραμμάτων και την αξιοπιστία

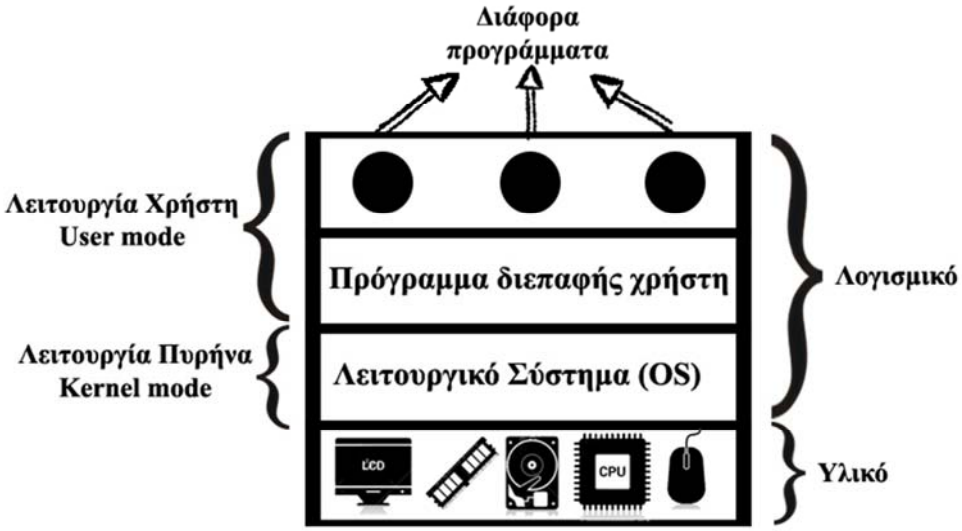
Αρκετοί άλλοι τύπου συνυπάρχουν με τους παραπάνω και παρέχουν και εκείνοι τα χαρακτηριστικά και τα εργαλεία για έναν ασφαλέστερο προγραμματισμό, όμως η επέκταση σε αυτές δεν είναι θέμα που πραγματεύεται η παρούσα διπλωματική.

1.3 Σύγχρονα λειτουργικά συστήματα

Ένας σύγχρονος υπολογιστής αποτελείται από έναν ή περισσότερους επεξεργαστές, την κύρια μνήμη, δίσκους, εκτυπωτές, πληκτρολόγιο, ποντίκι, μια οθόνη, τις διασυνδέσεις δικτύου, και διάφορες άλλες συσκευές εισόδου / εξόδου. Συνολικά είναι ένα πολύπλοκο σύστημα. Η διαχείριση όλων αυτών των στοιχείων και η χρήση τους κατά βέλτιστο τρόπο είναι ένα εξαιρετικά δύσκολο έργο. Για το λόγο αυτό, οι υπολογιστές είναι εφοδιασμένοι με ένα λογισμικό που ονομάζεται λειτουργικό σύστημα, του οποίου η δουλειά είναι να παρέχει προγράμματα ή εφαρμογές με έναν καλύτερο, απλούστερο, ξεκάθαρο τρόπο και να διαχειρίζεται όλους τους πόρους που μόλις αναφέραμε αποτελεσματικά.

Το πρόγραμμα με το οποίο οι χρήστες αλληλεπιδρούν συνήθως ονομάζεται κέλυφος (shell) όταν είναι βασισμένο σε κείμενο και GUI (Graphical User Interface)[1] όταν χρησιμοποιεί εικονίδια. Στην πραγματικότητα δεν είναι μέρος του λειτουργικού συστήματος, αν και χρησιμοποιεί το λειτουργικό σύστημα για να επιτελέσει το έργο του.

Μια απλή απεικόνιση των βασικών συστατικών ενός συστήματος, δίνεται στο Σχήμα 1.1. Βλέπουμε ότι στο κατώτερο επίπεδο βρίσκεται το υλικό, το οποίο αποτελείται από τα τσιπ, τους δίσκους, το πληκτρολόγιο, την οθόνη, και άλλα φυσικά αντικείμενα. Στην κορυφή του υλικού είναι το λογισμικό. Οι περισσότεροι υπολογιστές έχουν δύο τρόπους λειτουργίας: Τη λειτουργία πυρήνα (Kernel mode)[1] και τη λειτουργία χρήστη (User mode)[1]. Το λειτουργικό σύστημα είναι το πιο θεμελιώδες κομμάτι του λογισμικού και τρέχει σε λειτουργία πυρήνα (που ονομάζεται επίσης λειτουργία επόπτη ή διαχειριστή). Με αυτόν τον τρόπο έχει πλήρη πρόσβαση σε όλο τον εξοπλισμό και μπορεί να εκτελέσει οποιαδήποτε εντολή είναι ικανό το σύστημα να εκτελέσει. Το υπόλοιπο λογισμικό εκτελείται σε λειτουργία χρήστη, η οποία ουσιαστικά αποσκοπεί στον περιορισμό από την πρόκληση βλαβών στο σύστημα.



Σχήμα 1.1 Απλή απεικόνιση βασικών συστατικών συστήματος

Το πρόγραμμα διεπαφής χρήστη, κέλυφος (shell) ή GUI, είναι το χαμηλότερο επίπεδο λογισμικού στη λειτουργία χρήστη, και επιτρέπει στο χρήστη την εκκίνηση προγραμμάτων, όπως είναι ένα πρόγραμμα περιήγησης στο Web, έναν αναγνώστη e-mail, μία εφαρμογή αναπαραγωγής μουσικής κλπ. Τα προγράμματα αυτά κάνουν βαριά χρήση του λειτουργικού συστήματος με την έννοια ότι χρησιμοποιούν πολλούς πόρους και έτσι η διαθέσιμη μνήμη αλλά και οι επεξεργαστικές ικανότητες επιβαρύνονται σημαντικά. Η τοποθέτηση του λειτουργικού συστήματος φαίνεται στο Σχήμα 1.1. το οποίο παρέχει τη βάση για όλα τα άλλα λογισμικά.

Μια σημαντική διάκριση μεταξύ του λειτουργικού συστήματος και της κανονικής (λειτουργίας χρήστη), είναι ότι εάν σε έναν χρήστη δεν αρέσει ένα συγκεκριμένο πρόγραμμα ανάγνωσης e-mail, τότε είναι ελεύθερος να εγκαταστήσει ένα διαφορετικό ή να γράψει ένα δικό του, εφόσον το επιθυμεί. Δεν είναι όμως ελεύθερος να γράψει τον δικό του διαχειριστή ρολογιού, ο οποίος αποτελεί μέρος του λειτουργικού συστήματος και προστατεύεται από το υλικό κατά των προσπαθειών από τους χρήστες για τροποποίηση.

Σε πολλά συστήματα βέβαια υπάρχουν προγράμματα που εκτελούνται στη λειτουργία χρήστη, τα οποία όμως βοηθούν το λειτουργικό σύστημα ή εκτελούν κάποιες εργασίες διαχωρισμού. Για παράδειγμα, υπάρχει συχνά ένα πρόγραμμα που επιτρέπει στους χρήστες να αλλάξουν τους κωδικούς πρόσβασής τους. Το πρόγραμμα αυτό δεν αποτελεί μέρος του λειτουργικού συστήματος και δεν τρέχει σε λειτουργία πυρήνα, αλλά παρέχει μια ευαίσθητη λειτουργία και πρέπει να προστατεύεται με ειδικούς τρόπους.

Ένας βασικός κανόνας για το ποιά προγράμματα αποτελούν μέρος του λειτουργικού συστήματος είναι ο εξής: Ότι πρόγραμμα τρέχει σε λειτουργία πυρήνα είναι σαφώς μέρος του λειτουργικού συστήματος, όσα όμως προγράμματα είναι έξω από αυτή τη λειτουργία είναι επίσης μέρος του, ή τουλάχιστον συνδέονται στενά με το λειτουργικό σύστημα.

Τα λειτουργικά συστήματα διαφέρουν από τα προγράμματα του χρήστη. Ειδικότερα θα μπορούσαμε να πούμε πώς τα λειτουργικά συστήματα είναι τεράστια σε έκταση κώδικα, πολύ πιο πολύπλοκα, και μακρόβια. Ο πηγαίος κώδικας ενός λειτουργικού συστήματος, όπως του Linux ή των Windows είναι της τάξης των πέντε εκατομμυρίων γραμμών και συνέχεια μεγαλώνει. Για να φανταστούμε τι σημαίνει αυτό, πρέπει να σκεφτούμε την εκτύπωση πέντε εκατομμυρίων γραμμών σε μορφή βιβλίου.

Ένας από τους βασικότερους λόγους λοιπόν που τα λειτουργικά συστήματα αντέχουν στο χρόνο και χρησιμοποιούνται για μεγάλο χρονικό διάστημα, είναι και η πολυπλοκότητα που τα διακατέχει στην συγγραφή του κώδικά τους. Για παράδειγμα για να δημιουργηθεί η επόμενη έκδοση του slackware linux[2] περνάει ένα μεγάλο χρονικό διάστημα, διότι όπως εξηγήσαμε ο κώδικας αλλά και η πολυπλοκότητά της συγγραφής είναι δεδομένη.

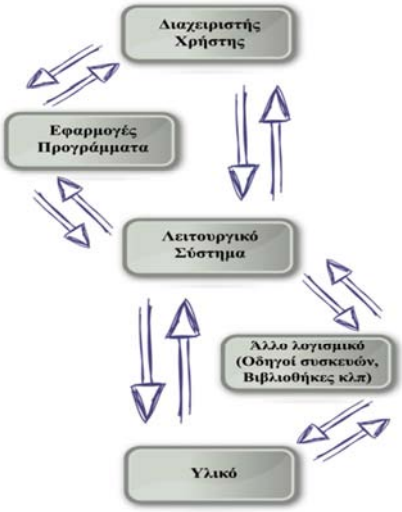
[1]Andrew S Tannenbaum, Κεφ.1ο, "Modern Operating Systems", Prentice Hall, 5th edition, 2009

[2]Συνήθως κάθε έναν χρόνο έχουμε και μία νέα έκδοση, αν και το τελευταίο χρονικό διάστημα δεν συμβαίνει αυτό με αποτέλεσμα η τελευταία έκδοση 14.1 του λειτουργικού συστήματος να είναι ενεργή πάνω από 3 χρόνια χωρίς να έχει ανακοινωθεί άλλη έκδοση), The Slackware Linux Project (http://www.slackware.com/)

1.3.1 Ορισμός λειτουργικού συστήματος

Ένα λειτουργικό σύστημα ("OS") είναι το πρόγραμμα που, αφού αρχικά φορτωθεί στον υπολογιστή από ένα πρόγραμμα εκκίνησης[1], διαχειρίζεται όλα τα άλλα προγράμματα[2] και τις συσκευές με τρόπο τέτοιο ώστε:

- ✓ Πολλαπλά προγράμματα να μπορούν να τρέχουν ταυτόχρονα.
- ✓ Το λειτουργικό σύστημα καθορίζει ποιες εφαρμογές θα πρέπει να τρέξουν, με ποια σειρά και πόσο χρόνο θα πρέπει να επιτρέπεται σε κάθε εφαρμογή να τρέξει, πριν δώσει σε μια άλλη εφαρμογή την προτεραιότητα.
- ✓ Διαχειρίζεται την κατανομή της εσωτερικής μνήμης μεταξύ των πολλαπλών εφαρμογών.
- ✓ Χειρίζεται τις εισόδους και εξόδους από και προς τις συνδεδεμένες συσκευές υλικού, όπως είναι οι σκληροί δίσκοι, εκτυπωτές, και οι πόρτες των dial-up.
- ✓ Στέλνει μηνύματα σε κάθε χρήστη (ή διαχειριστή συστήματος) σχετικά με την κατάσταση του λειτουργικού συστήματος και τυχόν λάθη που μπορεί να έχουν προκύψει.
- ✓ Σε υπολογιστές που μπορούν να παρέχουν παράλληλη επεξεργασία (πολλοί επεξεργαστές), ένα λειτουργικό σύστημα μπορεί να διαχειριστεί το πώς να διαιρέσει το πρόγραμμα έτσι ώστε να εκτελείται σε περισσότερους από έναν επεξεργαστές ταυτοχρόνως.



Σχήμα 1.2 Απλή απεικόνιση ροών ενός σύγχρονου λειτουργικού συστήματος

[1]Για να εκκινηθεί ένα σύστημα (αποκαλείται επίσης και ως "boot up") είναι απαραίτητο ένα πρόγραμμα αρχικής ουσιαστικά φόρτωσης του λειτουργικού συστήματος στην κύρια μνήμη του υπολογιστή ή της μνήμης τυχαίας προσπέλασης (RAM). Μόλις το λειτουργικό σύστημα φορτωθεί (για παράδειγμα, βλέπουμε την αρχική οθόνη της επιφάνειας εργασίας – εάν φυσικά χρησιμοποιείτε κάποια γραφική διεπαφή χρήστη (GUI)), και έτσι είναι έτοιμο προς τους χρήστες για να τρέξουν εφαρμογές.

[2]Τα άλλα προγράμματα ονομάζονται εφαρμογές ή προγράμματα εφαρμογών. Τα προγράμματα κάνουν χρήση του λειτουργικού συστήματος μέσω των αιτήσεων παροχής υπηρεσιών και διαμέσου μιας διεπαφής προγράμματος η οποία ορίζεται ως Διεπαφή Προγραμματισμού Εφαρμογών (API). Το API είναι ένας κώδικας που επιτρέπει σε δύο προγράμματα λογισμικού να επικοινωνούν μεταξύ τους. Επιπλέον, οι χρήστες μπορούν να αλληλεπιδρούν άμεσα με το λειτουργικό σύστημα, μέσω γλωσσικών εντολών ή μια γραφική διεπαφή χρήστη (GUI).

Ένας ορισμός που έχει δοθεί από τον **William Stallings** στο έργο του «**Operating Systems : Internals and Design Principles**» για το τι είναι ένα λειτουργικό σύστημα είναι ο εξής:

Ένα λειτουργικό σύστημα είναι ένα πρόγραμμα που ελέγχει την εκτέλεση των προγραμμάτων (εφαρμογών) και λειτουργεί ως μια διεπαφή μεταξύ των εφαρμογών και του υλικού στον υπολογιστή.

Τα λειτουργικά συστήματα έχουν τρεις στόχους όπως μας είπε:

- **Ευκολία:** Ένα λειτουργικό σύστημα κάνει έναν υπολογιστή πιο βολικό στη χρήση.
- **Αποδοτικότητα:** Ένα λειτουργικό σύστημα επιτρέπει στους πόρους του συστήματος να χρησιμοποιηθούν με αποτελεσματικό τρόπο.
- **Δυνατότητα να εξελίσσεται:** Ένα λειτουργικό σύστημα πρέπει να είναι κατασκευασμένο κατά τέτοιο τρόπο ώστε να επιτρέπει την αποτελεσματική ανάπτυξη, τη δοκιμή και εισαγωγή νέων λειτουργιών στο σύστημα.

Ενώ οι **Abraham Silberschatz, Peter Baer Galvin** και **Greg Gagne** στο έργο τους **Operating System Concepts** αναφέρουν πώς σε γενικές γραμμές, δεν έχουμε κανέναν απολύτως επαρκή ορισμό του τι είναι λειτουργικό σύστημα. Τα λειτουργικά συστήματα υπάρχουν επειδή προσφέρουν ένα λογικό τρόπο για την επίλυση του προβλήματος της δημιουργίας ενός χρησιμοποιήσιμου υπολογιστικού συστήματος. Ο θεμελιώδης στόχος των συστημάτων πληροφορικής είναι να εκτελούν τα προγράμματα των χρηστών και να κάνουν την επίλυση τους αντί για εκείνους.

Στο ίδιο περίπου συμπέρασμα καταλήγει και ο **Andrew S. Tanenbaum** του ανοικτού Πανεπιστημίου του Άμστερνταμ της Ολλανδίας στο έργο του **Modern Operating Systems** για το τι είναι λειτουργικό σύστημα, ο οποίος μας εξηγεί πώς:

Είναι δύσκολο να γίνει αντιληπτό και να πούμε τι είναι ένα λειτουργικό σύστημα εκτός από το να πούμε πώς είναι το λογισμικό που τρέχει στον πυρήνα λειτουργίας, και ακόμη και αυτός ο ορισμός δεν είναι πάντα αληθής.

Επίσης ο καθηγητής **Παύλος Σπυράκης** του **Ελληνικού Ανοικτού Πανεπιστημίου** μας εξηγεί:

Το λειτουργικό σύστημα είναι μια συλλογή από προγράμματα τα οποία ενεργούν ως «ενδιάμεσο» μεταξύ των χρηστών (π.χ. προγράμματα, εφαρμογές, συσκευές, άνθρωποι) και του Η/Υ. Χωρίς αμφιβολία, δύο είναι οι βασικοί στόχοι που εξυπηρετεί ένα λειτουργικό σύστημα:

- Να διαχειρίζεται τα δομικά στοιχεία (π.χ. μνήμη, hardware) του υπολογιστή
- Να εξασφαλίζει τη λειτουργία του Η/Υ με αποδοτικό τρόπο (ως προς το ποσοστό χρόνου χρήσης πόρων, τους χρόνους απόκρισης σε εντολές, τη διευκόλυνση του χρήστη κτλ.).

1.4 Ιστορική αναδρομή στα λειτουργικά συστήματα

Τα λειτουργικά συστήματα είχαν μία εξέλιξη στο χρόνο όπως και το υλικό των υπολογιστών που αναφέραμε στην παράγραφο 1.1. Με βάση αυτή την εξέλιξη η οποία χαρακτηρίζεται από σημαντικές αλλαγές ως προς τον τρόπο υλοποίησής τους, τους υπολογιστικούς πόρους αλλά και την αντιμετώπιση του τελικού χρήστη, μπορούμε να τα κατατάξουμε στις εξής κατηγορίες (**Σπυράκης** 2001) και χρονικές περιόδους:

1.4.1 Η μηδενική γενιά (1940)

Τα πρώτα υπολογιστικά συστήματα δεν είχαν λειτουργικό σύστημα. Οι χρήστες είχαν άμεση προσπέλαση στη γλώσσα μηχανής και προγραμμάτιζαν τα πάντα κυριολεκτικά «με το χέρι» (**Σπυράκης** 2001).

1.4.2 Η πρώτη γενιά (1950-1960)

Από το 1949 μέχρι το 1956 η βασική οργάνωση και ο τρόπος λειτουργίας του Η/Υ έμεινε σχετικά στο ίδιο σημείο. Η κλασική αρχιτεκτονική von Neumann των τότε Η/Υ συμπληρωνόταν με μια αυστηρά ακολουθιακή εκτέλεση εντολών, συμπεριλαμβάνοντας εντολές εισόδου – εξόδου. Ακόμα και στη διάρκεια του φορτώματος (loading) και του τρεξίματος προγραμμάτων, οι χρήστες δούλευαν πάνω στην «κονσόλα», μεταβάλλοντας το περιεχόμενο των καταχωρητών, εκτελώντας εντολές βήμα βήμα, εξετάζοντας θέσεις μνήμης και, γενικά, αλληλεπιδρώντας με τον Η/Υ στο χαμηλότερο δυνατό επίπεδο (μηχανής). Τα προγράμματα γράφονταν σε απόλυτη γλώσσα μηχανής (π.χ. δεκαδικό ή οκταδικό σύστημα) και για τη φόρτωσή τους χρησιμοποιούνταν ένας «απόλυτος» φορτωτής προγραμμάτων (loader), που ήταν σε καθορισμένη θέση της μνήμης και που φόρτωνε το πρόγραμμα σε κάποιο καθορισμένο σύνολο διαδοχικών θέσεων μνήμης. Σε εκείνα τα «άγρια χρόνια», το λεγόμενο «υποβοηθητικό λογικό» (programming aids) βασικά δεν υπήρχε. Σιγά σιγά, όμως, καθώς γινόταν αντιληπτή η σημασία του συμβολικού προγραμματισμού και καθώς οι μεταφραστές σε «γλώσσες μηχανής» (assemblers) άρχισαν να εμφανίζονται, άρχισε να δημιουργείται μια «προκαθορισμένη» σειρά λειτουργιών: Ένας φορτωτής φόρτωνε έναν assembler στο σύστημα. Ο assembler μετάφραζε (σε απόλυτο κώδικα μηχανής) μερικούς σωρούς από κάρτες με προγράμματα χρηστών και «λογικό βιβλιοθήκης» (library routines). Ο απόλυτος κώδικας που έβγαινε από αυτή τη φράση γραφόταν σε ταινία ή κάρτες και μετά ένας φορτωτής ξαναχρησιμοποιούνταν για να φορτώσει τον κώδικα (ως ένα ενιαίο σύνολο πια) στην κύρια μνήμη του (απόλυτου) προγράμματος. Οι παραπάνω αδυναμίες, μαζί με μια σειρά άλλους λόγους (το κόστος της συνεχούς ανθρώπινης παρέμβασης στη διαδικασία, τη διαθεσιμότητα της FORTRAN και άλλων γλωσσών, την ανάπτυξη κώδικα βιβλιοθήκης και κώδικα λειτουργιών εισόδου – εξόδου), έγιναν πιεστικοί λόγοι για την εξέλιξη των ΛΣ σε ΛΣ «πρώτης γενιάς». Τα πρώτα συστήματα batch αυτοματοποίησαν την ακολουθία:

φόρτωμα – μετάφραση – φόρτωμα – εκτέλεση, χρησιμοποιώντας ένα κεντρικό πρόγραμμα ελέγχου που ανακαλούσε και φόρτωνε «προγράμματα συστήματος» (system programms) (π.χ. assembler, compiler, φορτωτή ή κώδικες βιβλιοθήκης) και που αναλάμβανε τη μετάβαση από εργασία σε εργασία. Οι μεταφραστές γλωσσών ξαναγράφτηκαν, ώστε να παράγουν «μετατοπιζόμενο» (relocatable) κώδικα αντί για απόλυτο κώδικα. Οι «συνδετικοί φορτωτές» (linking loaders) άρχισαν να εμφανίζονται. Χάρη σε αυτούς, επιτράπηκε η ανάμειξη τμημάτων (ομάδων από κάρτες) «κώδικα πηγής» (source code) και τμημάτων μετατοπιζόμενου «κώδικα μηχανής αντικειμένου» (object code). Επίσης, οι κώδικες βιβλιοθήκης μπορούσαν πια να αποθηκεύονται σε μορφή μετατοπιζόμενου κώδικα. Σε εκείνα τα ΛΣ, τα θέματα προστασίας του συστήματος ήταν τα πιο δύσκολα προβλήματα. Ήταν σχετικά εύκολο για το σύστημα να καταστρέψει τον εαυτό του ή να καταστραφεί από κάποιο χρήστη ή ήταν πιθανό ένας χρήστης να «διαβάσει» πέρα από τη δική του εργασία, επεμβαίνοντας (ίσως άθελά του) στην επόμενη εργασία. Η κατανομή των πόρων (κύριας μνήμης και συσκευών Εισόδου/Εξόδου) ήταν δουλειά του χρήστη και όχι του ΛΣ. (**Tannenbaum** 2009 και **Σπυράκης** 2001)

1.4.3 Η δεύτερη γενιά (1959-1965)

Γύρω στα 1959 – 63 μερικές σημαντικές ανακαλύψεις στα συστήματα υλικού έδωσαν ώθηση στην παραπέρα εξέλιξη των ΛΣ. Ίσως η πιο σημαντική ανακάλυψη hardware, εκείνη την εποχή, ήταν το λεγόμενο «κανάλι δεδομένων» (data channel), δηλαδή ένας «πρωτόγονος» Η/Υ και τα εργαλεία εισόδου/εξόδου. Μόλις το κανάλι πάρει μια «εντολή αίτησης για I/O» από τον επεξεργαστή (ΚΜΕ), το κανάλι αρχίζει να εκτελεί και να ελέγχει την I/O εργασία ασύγχρονα και παράλληλα με την (συνεχιζόμενη εν τω μεταξύ) εκτέλεση εντολών από την ΚΜΕ. Δηλαδή η επικάλυψη των εργασιών της ΚΜΕ και των συστημάτων I/O (χρονικά) είναι πια γεγονός. Το κανάλι και η ΚΜΕ μοιράζονται την κύρια μνήμη, που περιέχει προγράμματα και δεδομένα και για τους δύο. Αρχικά, μόνο η ΚΜΕ μπορούσε να ρωτήσει ποια είναι η κατάσταση του καναλιού ανά πάσα στιγμή. Αργότερα όμως έγινε φανερό ότι το όλο σύστημα θα εργαζόταν πιο αποδοτικά εάν το κανάλι μπορούσε να διακόψει την εργασία της ΚΜΕ για να παραδώσει ένα μήνυμα, που συνήθως ήταν η λήξη μιας I/O εργασίας. Αμέσως άρχισαν να γράφονται πολύπλοκα συστήματα λογικού που μπορούσαν να εκμεταλλευτούν τις πιθανές χρήσεις της νέας αρχιτεκτονικής. Τα συστήματα αυτά περιλάμβαναν διαδικασίες «λογικής μόνωσης» (software buffering), που επέτρεπαν, για παράδειγμα, «στοίβαγμα» (queuing) αποτελεσμάτων λόγω καθυστέρησης γραψίματος της εισόδου κτλ. Ακόμα, περιλάμβαναν ρουτίνες «χειρισμού των σημάτων διακοπής» (interrupt handling), για να γίνεται δυνατή η διαδικασία της απάντησης (από την ΚΜΕ) σε περίπτωση I/O interrupt και η διαδικασία επιστροφής του ελέγχου στη διαδικασία που διακόπηκε, μετά την εξυπηρέτηση του σήματος διακοπής. (**Tannenbaum** 2009 και **Σπυράκης** 2001)

1.4.4 Η Τρίτη γενιά (1965-1980)

Η (φαινομενική) ασυμβατότητα ανάμεσα σε υπολογιστές για αριθμητικούς υπολογισμούς μεγάλης κλίμακας και σε εμπορικής χρήσης υπολογιστές αντιμετωπίστηκε πρώτα από την IBM με το σύστημα 360. Η γραμμή παραγωγής 360 ήταν μια σειρά από συμβατούς, όσον αφορά το λογισμικό, υπολογιστές. Το λειτουργικό τους σύστημα (OS 360), παρ' ότι μεγάλο και δύσχρηστο, εντούτοις εισάγει σημαντικότατες έννοιες, που δεν υπάρχουν στη δεύτερη γενεά. Μεταξύ αυτών, τα πλέον σημαντικά είναι αυτά του «πολυπρογραμματισμού» (multi programming). Η ιδέα χρησιμοποιεί διαχωρισμούς της μνήμης σε διάφορα «μέρη» (partitions), έτσι ώστε διάφοροι υπολογισμοί (εργασίες) να εξυπηρετούνται «ταυτό-χρονα». Με τον τρόπο αυτό κατέστη δυνατή η συνύπαρξη και η συνεκτέλεση αριθμητικών υπολογισμών μεγάλης κλίμακας και εμπορικών (εντατικών σε I/O) υπολογισμών για πρώτη φορά.

Άλλη σημαντική εξέλιξη των συστημάτων της τρίτης γενεάς υπήρξε η ικανότητα φόρτωσης πολλαπλών εργασιών (από κάρτες) στο σύστημα, ώστε να μην καθυστερεί η εκτέλεσή τους. Η ιδιότητα αυτή ονομάστηκε Spooling (από το Simultaneous Peripheral Operations on Line). Με τη χρήση του Spooling περιορίστηκε σοβαρά η χρήση μαγνητικών ταινιών. Η ανάγκη για γρήγορο χρόνο απόκρισης οδήγησε στην έννοια της «διαμοίρασης χρόνου» (time sharing), μια παραλλαγή του πολυπρογραμματισμού, όπου ο κάθε χρήστης έχει το δικό του τερματικό και το κεντρικό σύστημα μπορεί να δώσει «αλληλεπιδραστική» (interactive) εξυπηρέτηση σε όλους τους χρήστες χάρη στο μοίρασμα του χρόνου της CPU.

Το πλέον επιτυχημένο time–sharing λειτουργικό σύστημα της εποχής αυτής υπήρξε το MULTICS (διάδοχος του συστήματος CTSS, από το MIT, τα Bell Labs και τη General Electric και τα Unix), ικανό να εξυπηρετήσει εκατοντάδες χρήστες ταυτόχρονα. Οι ιδέες του MULTICS οδήγησαν το σχεδιαστή λειτουργικών συστημάτων Ken Thomson (των Bell Labs) στη δημιουργία του πρώτου UNIX συστήματος (μιας μικρής κλίμακας ειδικής των MULTICS για τη σειρά PDP). Το αρχικό του όνομα ήταν UNICS (Uniplexed Information and Computing Service), αλλά ο συν–σχεδιαστής του, ο B. Kernigham, το μετέτρεψε σε UNIX. Στην ομάδα προσετέθη και άλλος επιστήμονας των Bell Labs, ο D. Ritchie, και το UNIX ξαναγράφτηκε στη νέα (τότε) γλώσσα C (που σχεδιάστηκε και υλοποιήθηκε από τον Ritchie – αναφέρθηκε και στην ενότητα 1.2). Τα εργαστήρια Bell έδωσαν άδεια χρήσης και ανάπτυξης του UNIX (αναλυτικά στο κεφάλαιο 2) στα πανεπιστήμια δωρεάν και το σύστημα γρήγορα μετακόμισε σε πλειάδα αρχιτεκτονικών (π.χ. VAX, Motorola κτλ.). Η χρήση του ακόμα και σήμερα είναι ταχύτατα ανοδική. (**Tannenbaum** 2009 και **Σπυράκης** 2001)

1.4.5 Η τέταρτη γενιά (1980-1990)

Η τέταρτη γενιά λειτουργικών συστημάτων χαρακτηρίστηκε από την ταυτόχρονη εμφάνιση και κυριαρχία των προσωπικών υπολογιστών (PC's). Αυτή η πραγματικότητα οδήγησε στην εμφάνιση λειτουργικών συστημάτων «φιλικών προς το χρήστη» (user friendly) και ευέλικτων (π.χ. XENIX, DOS κτλ.) και αργότερα,

λόγω της εμφάνισης τοπικών δικτύων, σε ευέλικτα λειτουργικά συστήματα με αντιλήψεις κατανεμημένου υπολογισμού και πολλαπλών καθηκόντων (multi – tasking OS). (**Tannenbaum** 2009 και **Σπυράκης** 2001)

1.4.6 Η πέμπτη γενιά (1990-σήμερα)

Η σημερινή πραγματικότητα χαρακτηρίζεται από την κυριαρχία του Διαδικτύου και του Παγκόσμιου Ιστού (Internet και Web) και την ύπαρξη γρήγορων τοπικών και μη τοπικών δικτύων. Τα σύγχρονα λειτουργικά συστήματα τροποποίησαν τις παλαιότερες αρχές σχεδιασμού, ώστε να ικανοποιούν τη διαχείριση των δικτυακών πόρων. Έτσι, έχουμε την κυρίαρχη σχεδιαστική αντίληψη των «processes» (διεργασιών ή διαδικασιών), καθώς και τη σχεδιαστική αρχή της επικοινωνίας των διεργασιών με τη μέθοδο «Client – Server» (Πελάτη – Εξυπηρετητή) και άλλες μεθόδους επικοινωνίας (π.χ. sockets: συνδυασμός της ip με την εκάστοτε πόρτα), που συμβαδίζουν με τις τεχνολογικές εξελίξεις στα δίκτυα IP. Τα λειτουργικά συστήματα που κυριαρχούν χαρακτηρίζονται από την ανεξαρτησία τους από δεσμεύσεις του hardware, καθώς και από τη φιλικότητα προς το χρήστη και τις επικοινωνιακές τους ικανότητες. (**Tannenbaum** 2009 και **Σπυράκης** 2001)

Στις μέρες μας (2016), τα δημοφιλέστερα λειτουργικά συστήματα στους μικροϋπολογιστές, (συμπεριλαμβανομένων των προσωπικών υπολογιστών), έχουν διαμορφωθεί σε δύο μεγάλες οικογένειες: αυτή των Unix-συμβατών (unix-like) και την οικογένεια των Microsoft Windows. Οι κεντρικοί υπολογιστές και τα ενσωματωμένα συστήματα χρησιμοποιούν μια ποικιλία άλλων λειτουργικών συστημάτων, τα περισσότερα από τα οποία δεν έχουν άμεση συγγένεια με τα Windows ενώ κάποιες φορές είναι παρόμοια με το Unix. Τα *Unix-συμβατά* λειτουργικά συστήματα αποτελούν μια πολυποίκιλη ομάδα, με πολλές κύριες υποκατηγορίες συμπεριλαμβανομένων των System V[1], BSD[2], και GNU/Linux (αναλυτικά στο κεφάλαιο 3). Το εμπορικό σήμα Unix χρησιμοποιείται από πολλά λειτουργικά συστήματα που έχουν πολλά κοινά με το αρχικό Unix. Τα Unix λειτουργικά συστήματα τρέχουν σε μια μεγάλη γκάμα από αρχιτεκτονικές υπολογιστών. Χρησιμοποιούνται πολύ σαν συστήματα εξυπηρετητές στις επιχειρήσεις και σε σταθμούς εργασίας σε ακαδημαϊκούς και μηχανολογικούς χώρους εργασίας. Παραλλαγές του Unix που διακινούνται ως Ελεύθερο λογισμικό[3], όπως το GNU/Linux και BSD αυξάνουν σε δημοτικότητα στο χώρο των σταθμών εργασίας και των προσωπικών υπολογιστών λόγω της μεγάλης τους σταθερότητας έναντι των λειτουργικών συστημάτων της οικογένειας MS Windows. Παραλλαγές που διακινούνται με κλειστές άδειες χρήσης[4] όπως το HP-UX[5] της Hewlett-Packard, το Irix[6] της Silicon Graphics και το AIX[7] της IBM έχουν σχεδιαστεί να τρέχουν μόνο στο υλικό των συγκεκριμένων εταιρειών ενώ άλλες παραλλαγές μπορούν να τρέξουν και σε προσωπικούς υπολογιστές. Το Solaris[8] της Sun (που άλλαξε άδεια χρήσης σε λογισμικό ανοιχτού κώδικα υπό την CDDL[9] άδεια) είναι ένα τέτοιο πολύμορφο αλλά ουσιαστικά πρόκειται για καθαρό Unix και μπορεί να τρέχει στους σταθμούς εργασίας της Sun αλλά και στον μικρότερο υπολογιστή αρχιτεκτονικής x86. Το λειτουργικά συστήματα Mac OS X[10] της Apple είναι μια παραλλαγή του BSD, και έχει αντικαταστήσει τα προηγούμενα (μη-unix) Mac OS λειτουργικά συστήματα της Apple σε μια σχετικά μικρή αλλά αφοσιωμένη αγορά, έχοντας γίνει ένα πολύ

δημοφιλές Unix. Η οικογένεια των Microsoft Windows ξεκίνησε σαν ένα επίπεδο γραφικής διασύνδεσης πάνω από το παλιότερο MS-DOS[11] περιβάλλον για τους IBM υπολογιστές. Οι σύγχρονες εκδόσεις των Windows βασίζονται στον καινούργιο πυρήνα των Windows NT που πρωτοδιαμορφώθηκε στο OS/2[12], με πιο πρόσφατη έκδοση εν έτη 2015 τα Windows 10. Τα Windows τρέχουν πάνω σε 32- και 64-bit Intel και AMD υπολογιστές, αν και προηγούμενες εκδόσεις έτρεχαν και σε DEC Alpha, MIPS και PowerPC αρχιτεκτονικές. (**Wikipedia**, 2016)

---

[1]Το System V ή όπως αποκαλείται αλλιώς σύστημα πέντε είναι μία από τις πρώτες εμπορικές εκδόσεις του λειτουργικού συστήματος Unix.

[2]Το BSD (Berkeley Software Distribution - μερικές φορές αναφέρεται και ως *Berkeley Unix*) είναι ένα σύστημα προερχόμενο από το Unix, που διανέμεται από το πανεπιστήμιο του Μπέρκλεϊ της Καλιφόρνιας. Αναλυτικότερα στην ενότητα 2.3.6.

[3]Το Ελεύθερο Λογισμικό, όπως ορίζεται στον Ορισμό του Ελεύθερου Λογισμικού από το Ίδρυμα Ελευθέρου Λογισμικού (Free Software Foundation), είναι λογισμικό που μπορεί να χρησιμοποιηθεί, αντιγραφεί, μελετηθεί, τροποποιηθεί και αναδιανεμηθεί χωρίς περιορισμό.

[4]Μια άδεια κλειστού λογισμικού επιτρέπει στον τελικό χρήστη την χρήση του λογισμικού εντός περιορισμένου περιβάλλοντος, απαγορεύοντας την αποσυμπίληση του λογισμικού (δηλαδή το να επιχειρηθεί η ανάκτηση του πηγαίου κώδικα από αυτή του κώδικα μηχανής), και την ανάλυση, τροποποίηση και αναδημιουργία του λογισμικού, και συνήθως απαγορεύονται επίσης η αντιγραφή και διανομή του λογισμικού (είτε δωρεάν είτε επί πληρωμή) από τρίτα πρόσωπα, καθώς και η χρήση του από ένα μεγάλο αριθμό ανθρώπων ή η εγκατάστασή του σε πολλούς υπολογιστές.

[5]Το HP-UX (Hewlett-Packard Unix) είναι μία ιδιόκτητη εφαρμογή της Hewlett-Packard του λειτουργικού συστήματος Unix, βασισμένο στο UNIX System V και κυκλοφόρησε για πρώτη φορά το 1984

[6]Το IRIX είναι ένα λειτουργικό σύστημα που αναπτύχθηκε από την εταιρεία Silicon Graphics Inc για να χρησιμοποιηθεί στους ηλεκτρονικούς υπολογιστές της (workstations και servers). Είναι βασισμένο στο λειτουργικό σύστημα UNIX System V με κάποιες επεκτάσεις από το BSD

[7]Το AIX (Advanced Interactive Executive) είναι μια σειρά από ιδιόκτητα Unix συστήματα τα οποία αναπτύσσονται και πωλούνται από την IBM για αρκετούς από τους υπολογιστές της

[8]Το Solaris, είναι ένα λειτουργικό σύστημα βασισμένο στο UNIX και σχεδιασμένο από την εταιρία Sun Microsystems. Η άδεια χρήσης του Solaris είναι κλειστή, αλλά πρόσφατα μεγάλα κομμάτια του συστήματος έχουν αλλάξει άδεια χρήσης σε ανοιχτού κώδικα (CCDL) διαμορφώνοντας το OpenSolaris.

[9]Η Common Development and Distribution License (CDDL) είναι μια άδεια χρήσης ελεύθερου λογισμικού, που δημιουργήθηκε από την Sun Microsystems, η οποία βασίζεται στο Mozilla Public License (MPL)

[10]Το OS X, πρώην Mac OS X είναι μια σειρά γραφικών λειτουργικών συστημάτων που αναπτύσσεται, προωθείται και πωλείται από την Apple Inc. και συμπεριλαμβάνεται σε κάθε καινούριο υπολογιστή Macintosh (Mac). Το Mac OS X είναι η εξέλιξη του αρχικού Mac OS το οποίο ήταν το αρχικό λειτουργικό σύστημα της Apple την περίοδο 1984-1999. Σε αντίθεση με το αρχικό Mac OS, το Mac OS X είναι ένα UNIX λειτουργικό σύστημα το οποίο άρχισε να αναπτύσσεται στην εταιρία NeXT από τα τέλη του 1980 και μέχρι την εξαγορά της από την Apple το 1997.

[11]Το MS-DOS (MicroSoft-Disk Operating System), είναι λειτουργικό σύστημα με περιβάλλον γραμμής εντολών που δημιουργήθηκε από την Microsoft. Ήταν το πιο διαδεδομένο μέλος της οικογένειας των λειτουργικών συστημάτων DOS. Την δεκαετία του 1980 ήταν το επικρατέστερο λειτουργικό σύστημα για την συμβατή πλατφόρμα προσωπικών υπολογιστών. Βαθμιαία αντικαταστάθηκε από διάφορες εκδόσεις του λειτουργικού συστήματος Windows.

[12]Το OS/2 είναι μια σειρά από λειτουργικά συστήματα, που αρχικά δημιουργήθηκαν από τη Microsoft και την IBM, αργότερα αναπτύχθηκε από την IBM αποκλειστικά. Η πρώτη έκδοση του OS / 2 κυκλοφόρησε τον Δεκέμβριο του 1987 και νεότερες εκδόσεις κυκλοφόρησαν μέχρι τον Δεκέμβριο του 2001. Το OS/2 δεν διανέμετε πλέον στην αγορά από την IBM, και η υποστήριξή του από την IBM διεκόπη στις 31 Δεκεμβρίου 2006, έκτοτε υποστηρίζεται και αναβαθμίζεται με μερικά νέα χαρακτηριστικά κάτω από το όνομα eComStation.

1.5 Δομή ενός λειτουργικού συστήματος

Η βασική δομή ενός λειτουργικού συστήματος (**Κατσούλας** κ.α., **Silberschatz** et al, 2009) αποτελείτε από τα εξής υποσύνολα:

α) Τον Πυρήνα (Kernel). Είναι το κυριότερο τμήμα ενός ΛΣ. Το τμήμα αυτό φορτώνεται πρώτο στην κύρια μνήμη και εκτελείται συνεχώς σε όλη τη διάρκεια λειτουργίας του υπολογιστή. Τα προγράμματα εφαρμογών επικοινωνούν με αυτό μέσα από ένα καθορισμένο σύνολο κλήσεων. Ο πυρήνας είναι ο κύριος υπεύθυνος για τη συνεργασία του λογισμικού με το υλικό του υπολογιστή,
β) Το Σύστημα Αρχείων (File System). Είναι το τμήμα του ΛΣ το οποίο διαχειρίζεται τα αρχεία (ονοματοδοσία, καταχώριση, ανάκτηση κ.λπ.) και φροντίζει επίσης για τη διάθεσή τους στους χρήστες, Το σύστημα αρχείων έχει ιδιαίτερο ενδιαφέρον. Τα προγράμματα προφανώς χρειάζονται να διαβάσουν, να εγγράψουν αρχεία και καταλόγους. Επίσης χρειάζονται να δημιουργήσουν και να διαγράψουν αρχεία με βάση το όνομα, να ψάξουν για ένα συγκεκριμένο αρχείο ή καταλόγους με αρχεία πληροφοριών. Τέλος, ορισμένα προγράμματα περιλαμβάνουν τα δικαιώματα διαχειριστή, για να επιτρέπουν ή να αρνούνται την πρόσβαση σε αρχεία ή καταλόγους.
γ) Τη διεπαφή χρήστη (User Interface) ή το κέλυφος (Shell). Είναι το τμήμα που αναλαμβάνει να δέχεται και να δίνει στο σύστημα του υπολογιστή τα αιτήματα (εντολές) του χρήστη και επίσης να μεταφέρει στο χρήστη μηνύματα από το σύστημα. Το τμήμα αυτό δημιουργεί το περιβάλλον επικοινωνίας χρήστη – υπολογιστή και μπορεί να υλοποιηθεί με *περιβάλλον γραμμής εντολών* ή με *γραφικό περιβάλλον* ή και με τους δύο τρόπους. διεπαφή χρήστη.
Σχεδόν όλα τα λειτουργικά συστήματα έχουν μια διεπαφή χρήστη (UI). Αυτή η διεπαφή μπορεί να λάβει διάφορες μορφές. Η μία είναι μια διεπαφή γραμμής εντολών (CLI), η οποία χρησιμοποιεί εντολές κειμένου και μια μέθοδο για την εισαγωγή τους (ας πούμε, ένα πρόγραμμα για να επιτρέπει την εισαγωγή και επεξεργασία των εντολών). Μία άλλη είναι μια bach διεπαφή, στην οποία οι εντολές και οδηγίες για τον έλεγχο των εντολών καταχωρούνται σε αρχεία, και τα αρχεία αυτά εκτελούνται (ουσιαστικά πρόκειται για scripts τα οποία έχουν προκαθορισμένες εντολές προς εκτέλεση). Ενώ συνηθέστερα, χρησιμοποιείται μία γραφική διεπαφή χρήστη (GUI). Μερικά συστήματα παρέχουν δύο ή και τις τρείς από αυτές τις δυνατότητες. Στις μέρες μας όλα τα σύγχρονα λειτουργικά συστήματα παρέχουν τα παραπάνω

1.5.1 Γενική ανάλυση του πυρήνα (kernel)

Όλες οι λειτουργίες που αφορούν τις διαδικασίες υλοποιούνται από το μέρος του λειτουργικού συστήματος που ονομάζεται «πυρήνας» (kernel). Ο πυρήνας αποτελεί μόνο ένα μικρό κλάσμα ολόκληρου του λειτουργικού συστήματος (**Σπυράκης**, 2001), όμως είναι από τα περισσότερο χρησιμοποιούμενα μέρη του.
Επίσης, επειδή ο πυρήνας χτίζεται κατευθείαν πάνω στο υλικό του Η/Υ, είναι το περισσότερο εξαρτώμενο από τη μηχανή μέρος του λειτουργικού συστήματος και είναι απαραίτητο να γραφεί σε «γλώσσα μηχανής» (assembly), με λίγες μικρές εξαιρέσεις. Ο περιορισμός στη χρησιμοποίηση της γλώσσας assembly σε ένα μικρό

μέρος του λειτουργικού συστήματος (π.χ. στο UNIX όχι παραπάνω από 1000 εντολές) έγινε δυνατός χάρη στην ιεραρχική δομή των λειτουργικών συστημάτων και προσέφερε αρκετά στην υλοποίηση ενός προϊόντος απαλλαγμένου από λάθη, εύκολα κατανοήσιμου και ≪επιδεκτικού συντήρησης≫ (maintainable).

Ο πυρήνας ενός λειτουργικού συστήματος συνήθως περιέχει κώδικα που υλοποιεί τις παρακάτω λειτουργίες:

1. **Χειρισμός διακοπών**

Σύμφωνα με τον καθηγητή **Σπυράκη** το λειτουργικό σύστημα συμπεριλαμβάνει ρουτίνες γνωστές ως ≪χειριστές διακοπών≫, που είναι υπεύθυνες για την επεξεργασία κάθε διαφορετικού τύπου διακοπής. Τρεις είναι οι βασικές ενέργειες που πρέπει να γίνουν για την εξυπηρέτηση μιας διακοπής:

- ✓ Η διάσωση της κατάστασης της διακοπτόμενης διαδικασίας.
- ✓ Ο καθορισμός της πηγής της διακοπής.
- ✓ Η μεταβίβαση του ελέγχου στον αντίστοιχο χειριστή της διακοπής για την εξυπηρέτηση της διακοπής.

Οι παραπάνω τρεις λειτουργίες υλοποιούνται από το μέρος του λειτουργικού συστήματος που ονομάζεται ≪χειριστής διακοπών πρώτου επιπέδου≫ (ΧΕΔΠΕ). Ο ≪χειριστής διακοπών πρώτου επιπέδου≫ είναι συνήθως ένα σχετικά απλό πρόγραμμα που λειτουργεί σε ένα συγκεκριμένο μέρος της μνήμης, το δε σύνολο των καταχωρητών που απαιτείται για τη λειτουργία του δεν είναι μεγάλο.

2. **Δημιουργία και καταστροφή διαδικασιών**

Η διαδικασία είναι μία σειρά προγραμματιστικών εντολών με συγκεκριμένο «σημείο εισόδου εκτέλεσης» και ένα ή περισσότερα «σημεία εξόδου», δηλαδή εντολές διακλάδωσης σε άλλο τμήμα του ευρύτερου εκτελούμενου προγράμματος (**Wheeler**, 1952).

3. **Δρομολόγηση (dispatching)**

Η ταχύτητα δρομολόγησης είναι ένα ≪κρίσιμο≫ σημείο για το λειτουργικό σύστημα, και γι’ αυτό πρέπει να διατηρείται υψηλή.

4. **Νάρκωση και αφύπνιση διαδικασίας**

Μια διαδικασία μπορεί να βρεθεί στην κατάσταση ≪ναρκωμένη–έτοιμη≫ και ≪ναρκωμένη–μπλοκαρισμένη≫. Όταν αφυπνιστεί μια ≪ναρκωμένη–έτοιμη≫ διαδικασία, γίνεται ≪έτοιμη≫. Όταν μια διαδικασία είναι ≪τρέχουσα≫ και ναρκωθεί, γίνεται ≪ναρκωμένη–έτοιμη≫, ενώ, όταν είναι ≪μπλοκαρισμένη≫, γίνεται ≪ναρκωμένη–μπλοκαρισμένη≫. Η ≪ναρκωμένη–μπλοκαρισμένη≫ διαδικασία για την οποία έχει περατωθεί η αιτία για την οποία μπλοκαρίστηκε θα γίνει ≪ναρκωμένη–έτοιμη≫. Οι λόγοι που οδηγούν στη νάρκωση μια ≪μπλοκαρισμένη≫ διαδικασία είναι το γεγονός ότι η αιτία του μπλοκαρίσματος είναι δυνατόν να αργήσει απεριόριστα.

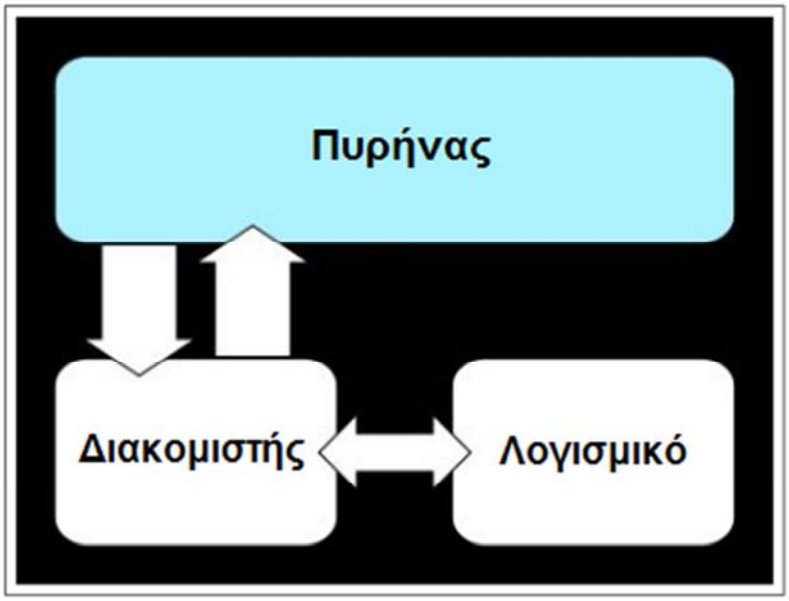
5. **Συντονισμός διαδικασιών**

Αφορά την ανάγκη για τον συντονισμό δύο διαδικασιών οι οποίες τρέχουν ταυτόχρονα σε ένα σύστημα και έχουν κοινές μεταβλητές. Η αλλαγή των κοινών μεταβλητών μπορεί να επηρεάσει την ταχύτητα εκτέλεσης των δύο διαδικασιών.

6. **Επικοινωνία μεταξύ διαδικασιών**
7. **Υποστήριξη δραστηριοτήτων Ε/Ε**
8. **Υποστήριξη δέσμευσης και αποδέσμευσης μνήμης κτλ.**

1.5.1.1 Αρχιτεκτονικές πυρήνα και διαφορές

1. **Ο microkernel**



Σχήμα 1.3 Αναπαράσταση microkernel πυρήνα (πηγή: Bitterling, 2009)

Σύμφωνα με τον **Patrick Bitterling** του **Πανεπιστημίου του Βερολίνου** ο σχεδιασμός ενός microkernel πυρήνα προσδιορίζει ότι έχει λειτουργίες μόνο για μια απλή επικοινωνία μεταξύ των διεργασιών, της διαχείρισης μνήμης και τον προγραμματισμό. Όλες αυτές οι λειτουργίες διεξάγονται στην λειτουργία πυρήνα, αλλά οι υπόλοιπες τρέχουν σε λειτουργία χρήστη. Ο microkernel δεν υλοποιείτε ως μία τεράστια διαδικασία. Η λειτουργικότητα του microkernel χωρίζεται σε διάφορες διεργασίες που ονομάζονται Servers. Στην καλύτερη περίπτωση μόνο αυτοί οι διακομιστές (Servers) μπορούν να πάρουν περισσότερα προνόμια τα οποία τους χρειάζονται για να εκτελέσουν τις διάφορες εργασίες που τους έχουν ανατεθεί. Όλοι οι διακομιστές διαχωρίζονται από το σύστημα και κάθε διεργασία έχει το δικό της χώρο διευθύνσεων[1] (address space). Το αποτέλεσμα είναι ότι ο microkernel δεν

μπορεί να ξεκινήσει άμεσα τις λειτουργίες του. Θα πρέπει να επικοινωνήσει μέσω ενός "Αποστελλόμενου Μηνύματος (Message Passing)", το οποίο είναι ένας μηχανισμός επικοινωνίας μεταξύ διεργασιών και που επιτρέπει στους Servers να επικοινωνούν μεταξύ τους. Εξαιτίας αυτής της εφαρμογής τα σφάλματα επηρεάζουν μόνο τη διαδικασία στην οποίο εντάσσονται. Αυτές οι τμηματοποιήσεις επιτρέπουν την ανταλλαγή των διακομιστών χωρίς την εμπλοκή όλου του συστήματος. Η επικοινωνία μέσω των αποστελλόμενων μηνυμάτων παράγει περισσότερες εξόδους, παρά μια κλήση συνάρτησης και περισσότερες ανταλλαγές περιεχομένου (context switches[2]) από έναν μονολιθικό πυρήνα. Το αποτέλεσμα των context switches και οι πολλές ανταλλαγές περιεχομένου δια μέσω των απεσταλμένων μηνυμάτων είναι μια καθυστέρηση (latency), η οποία οδηγεί σε μια αρνητική απόδοση.

Πλεονεκτήματα microkernel πυρήνα:

- ✓ Η συντήρηση είναι γενικά ευκολότερη σε σχέση με τον μονολιθικό πυρήνα.
- ✓ Τα Patches μπορούν να δοκιμαστούν σε ξεχωριστά μέρη, και στη συνέχεια εκχωρηθούν στον κώδικα.
- ✓ Το νέο λογισμικό μπορεί να ελεγχθεί χωρίς να χρειάζεται η επανεκκίνηση του πυρήνα.
- ✓ Περισσότερα δυναμικότητα σε γενικές γραμμές

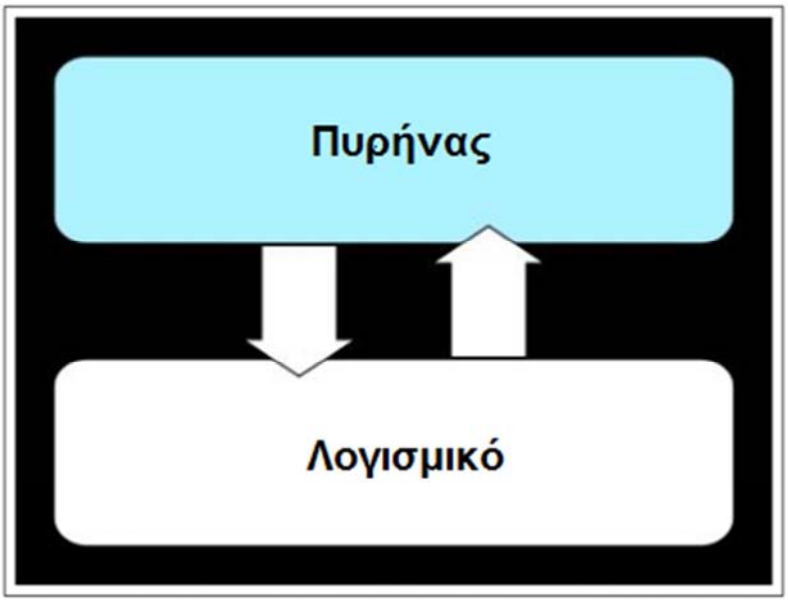
Μειονεκτήματα microkernel πυρήνα:

- ✓ Μεγαλύτερη κατάχρηση της μνήμης κατά την λειτουργία
- ✓ Απαιτείτε περισσότερο λογισμικό για τις διάφορες διασύνδεσης, οπότε έχουμε απώλεια απόδοσης.
- ✓ Τα μηνύματα σφαλμάτων μπορεί να είναι πιο δύσκολο να διορθωθούν, λόγω των μεγαλύτερων κύκλων στα κυκλώματα που πρέπει να κάνουν έναντι του μονολιθικού πυρήνα.
- ✓ Η διαχείριση διαδικασιών σε γενικές γραμμές μπορεί να είναι πολύ περίπλοκη

[1]Ορίζει μια σειρά διακριτών διευθύνσεων, κάθε μία από τις οποίες μπορούν να αντιστοιχούν σε έναν host δικτύου, σε μία περιφερειακή συσκευή, έναν τομέα δίσκου, ένα κελί μνήμης ή άλλη λογική ή φυσική οντότητα.

[2]Είναι η διαδικασία της αποθήκευσης και της αποκατάστασης μιας κατάστασης (πιο συγκεκριμένα, το πλαίσιο εκτέλεσης) μιας διαδικασίας ή ενός νήματος, έτσι ώστε η εκτέλεση να μπορεί να συνεχιστεί από το ίδιο σημείο σε μεταγενέστερο χρόνο.

2. Ο μονολιθικός πυρήνας (monolithic)



Σχήμα 1.4 Αναπαράσταση μονολιθικού πυρήνα (πηγή: Bitterling, 2009)

Σε αντίθεση με τον microkernel ο μονολιθικός πυρήνας περιλαμβάνει περισσότερες λειτουργίες (**Bitterling,** 2009**)**. Έτσι υπάρχουν περισσότερες υπηρεσίες που εκτελούνται σε λειτουργία πυρήνα όπως είναι όλα τα προγράμματα οδήγησης συσκευών, ο dispatcher, ο χρονοπρογραμματισμός, η εικονική μνήμη, όλες οι επικοινωνίες μεταξύ των διεργασιών (όχι μόνο η απλή IPC όπως στο microkernel), το (εικονικό) σύστημα αρχείων και οι κλήσεις του συστήματος, έτσι ώστε μόνο οι εφαρμογές/προγράμματα να τρέχουν σε λειτουργία χρήστη. Ο μονολιθικός πυρήνας υλοποιείται ως μία και μόνο διαδικασία, η οποίο τρέχει σε έναν ενιαίο χώρο διευθύνσεων.

Όλες οι υπηρεσίες πυρήνα εκτελούνται σε έναν χώρο διευθύνσεων, έτσι η επικοινωνία μεταξύ των υπηρεσιών αυτών είναι πιο απλή, επειδή ο πυρήνας έχει την ικανότητα να επεξεργάζεται όλες τις λειτουργίες άμεσα όπως ένα πρόγραμμα στην λειτουργία χρήστη.

Η ικανότητα της εκτέλεσης των κλήσεων συστήματος έχει ως αποτέλεσμα την καλύτερη απόδοση και απλούστερη εκτέλεση των λειτουργιών του πυρήνα. Η συντριβή (crash) ή ένα σφάλμα (bug) σε μια μονάδα που λειτουργεί σε λειτουργία πυρήνα μπορεί να συντρίψει ολόκληρο το σύστημα

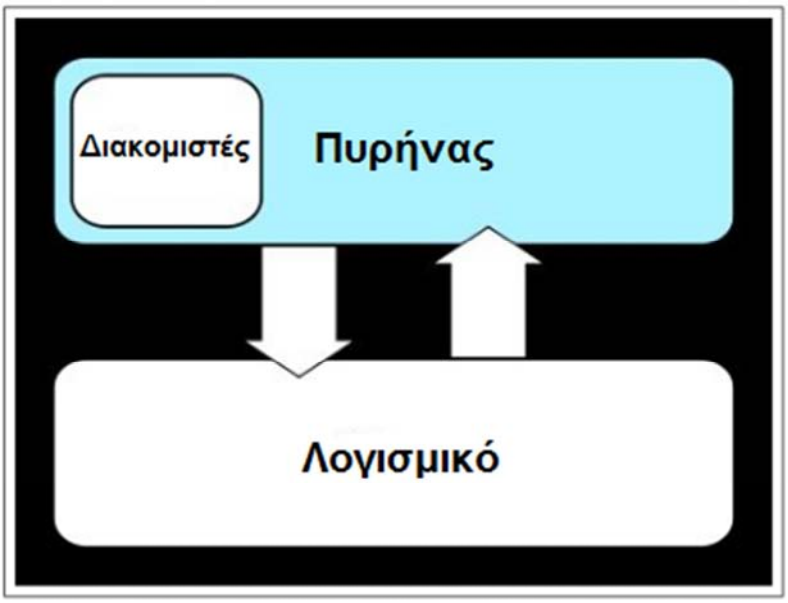
Πλεονεκτήματα μονολιθικού πυρήνα:

- ✓ Επειδή υπάρχει λιγότερο λογισμικό που εμπλέκεται (απουσία ενδιάμεσων επικοινωνιών όπως στον microkernel με τον Server) είναι πιο γρήγορος
- ✓ Καθώς πρόκειται για ένα ενιαίο κομμάτι λογισμικού, είναι μικρότερος τόσο στην πηγή (source) αλλά και στις compiled μορφές.
- ✓ Λιγότερος κώδικας γενικά σημαίνει λιγότερα σφάλματα (bugs), το οποίο μπορεί να μεταφραστεί σε λιγότερα προβλήματα ασφάλειας.

Μειονεκτήματα μονολιθικού πυρήνα:

- ✓ Εξαρτήσεις μεταξύ των στοιχείων του συστήματος
- ✓ Ένα σφάλμα σε ένα πρόγραμμα οδήγησης θα μπορούσε να συντρίψει ολόκληρο το σύστημα
- ✓ Πολυπλοκότητα στην συντήρηση και αναβάθμιση του πυρήνα αφού πρόκειται για μία ενιαία μορφή.

3. Ο υβριδικός πυρήνας (hybrid or modular)



Σχήμα 1.5 Αναπαράσταση υβριδικού πυρήνα (πηγή: Bitterling, 2009)

Ο υβριδικός πυρήνας είναι κάτι μεταξύ του microkernel και του μονολιθικού πυρήνα (που αυτός είναι και ο λόγος για το όνομα του). Ο υβριδικός πυρήνας τρέχει τις ίδιες διεργασίες στη λειτουργία πυρήνα όπως ο microkernel. Επιπροσθέτως ο υβριδικός πυρήνας τρέχει την IPC (InterProcess Communication, διευκόλυνση της ανταλλαγής δεδομένων και του συγχρονισμού μεταξύ των ταυτοχρόνως εκτελούμενων διεργασιών) εφαρμογή και τους οδηγούς συσκευών σε λειτουργία πυρήνα. Η λειτουργία χρήστη χρησιμοποιείται για τον UNIX-Server, File-Server και τις εφαρμογές. Ο στόχος αυτής της αρχιτεκτονικής είναι να πάρει τα οφέλη επίδοσης ενός μονολιθικού πυρήνα, με την σταθερότητα ενός microkernel. Το αποτέλεσμα αυτής της αρχιτεκτονικής είναι μία παρόμοια με την microkernel δομή που μοιάζει με τον μονολιθικό πυρήνα. (**Bitterling,** 2009**)**

Πλεονεκτήματα υβριδικού πυρήνα:

- ✓ Ταχύτερος χρόνος ανάπτυξης για τους οδηγούς. Δεν απαιτείται επανεκκίνηση για τη δοκιμή.
- ✓ Ικανότητα άμεσων αιτημάτων σε αντιδιαστολή με την δαπάνη χρόνου για recompiling ολόκληρου του πυρήνα, για τα θέματα όπως νέους οδηγούς οδήγησης ή υποσυστήματα.
- ✓ Ταχύτερη ενσωμάτωση τεχνολογίας τρίτων προμηθευτών και κατασκευαστών

Μειονεκτήματα υβριδικού πυρήνα:

- ✓ Περισσότερες διασυνδέσεις οδηγούν σε αυξημένη πιθανότητα σφαλμάτων (bags), το οποίο μεταφράζεται σε περισσότερα κενά ασφαλείας.
- ✓ Η συντήρηση των modules μπορεί να προκαλέσει σύγχυση σε μερικούς διαχειριστές όταν ασχολούνται με προβλήματα όπως οι διαφορές μεταξύ των συμβόλων

1.5.2 Γενική ανάλυση του συστήματος αρχείων (File System)

' Ένα σύστημα αρχείων (file system) αποτελεί τον τρόπο οργάνωσης και διαχείρισης της πληροφορίας που αποθηκεύεται σε ένα μέσο (δισκέτα, σκληρός δίσκος, μαγνητική ταινία κλπ) και, ειδικότερα, σε μία σεσκευή block. Οι επιλογές που οφείλει να κάνει κανείς κατά το σχεδιασμό ενός συστήματος αρχείων είναι πολλές και επηρεάζουν σημαντικά τις επιδόσεις του τελικού συστήματος. Κάποιες από τις επιλογές αυτές αφορούν την εσωτερική οργάνωση της πληροφορίας (δομές δεδομένων, αλγόριθμοι διαχείρισης των blocks κλπ) ενώ άλλες σχετίζονται με τις ιδιαιτερότητες του συστήματος στο οποίο θα εγκατασταθεί και την αλληλεπίδραση του με τα άλλα υποσυστήματα (πυρήνα, συσκευές). Όπως και σε πολλές άλλες αντίστοιχες περιπτώσεις, οι αποφάσεις που πρέπει να λάβει κανείς για τη βελτίωση των χαρακτηριστικών και της επίδοσης ενός συστήματος αρχείων είναι πολλές φορές αντιφατικές, με αποτέλεσμα να μην μπορεί κανείς να μιλήσει για το "τέλειο" σύστημα αρχείων αλλά για την καταλληλότερη επιλογή σε σχέση με τις ανάγκες της εκάστοτε εφαρμογής.

Κάποια από τα περισσότερο διαδεδομένα συστήματα αρχείων σήμερα είναι τα UFS (Unix File System), ext2,ext3, ext4 (Second, third and fourth extended Fle System) τα οποία είναι και τα προεπιλεγμένα σχεδόν σε όλες τις εκδόσεις των Linux, NTFS, fat και vfat (από τη Microsoft), XFS (από την SGI), JFS (από την IBM**),** ReiserFS, minix fs κ.α. Στα παραδείγματα που αναφέρθηκαν περιλαμβάνονται και οι τρεις περισσότερο διαδεδομένοι τύποι των συστημάτων αρχείων: αυτά που ακολουθούν τη νοοτροπία του UNIX (π.χ. UFS, ext2, ext3, ext4), τα journaling (XFS, ΈE, ReiserFS) και η προσέγγιση της Microsoft (NTFS, vfat). Οι τρεις αυτοί διαφορετικοί τύποι διαφέρουν σημαντικά μεταξύ τους στη φιλοσοφία λειτουργίας, τον τρόπο οργάνωσης της πληροφορίας, τις μεθόδους αλληλεπίδρασης με τις υπόλοιπες μονάδες του συστήματος, τις δυνατότητες που παρέχουν στο χρήστη αλλά και τον τρόπο παρουσίασης της πληροφορίας και αλληλεπίδρασης μαζί του.

Κύρια λειτουργία ενός συστήματος αρχείων είναι η αποθήκευση πληροφορίας σε αρχεία με τρόπο οργανωμένο, συνεπή και αξιόπιστο. Παρά το γεγονός ότι η επίτευξη του στόχου αυτού απαιτεί πολλές εσωτερικές λειτουργίες, δομές και ρουτίνες, στον τελικό χρήστη η συνολική πληροφορία φαίνεται οργανωμένη σε αρχεία, ενώ όλες οι υπόλοιπες (διαχειριστικές) εσωτερικές δομές δεν του είναι εμφανείς ή άμεσα προσπελάσιμες. (**Ευσταθόπουλος**, 2001)

Λειτουργικό σύστημα	NTFS	HFS+	FAT32	Ext2/Ext3/Ext4
Linux	Ανάγνωση/ Εγγραφή	Ανάγνωση/ Εγγραφή	Ανάγνωση/ Εγγραφή	Ανάγνωση/ Εγγραφή
Windows	Ανάγνωση/ Εγγραφή	Ανάγνωση/ Εγγραφή	Ανάγνωση/ Εγγραφή	-
Mac OS X	Ανάγνωση/ Εγγραφή	Ανάγνωση/ Εγγραφή	Ανάγνωση/ Εγγραφή	Ανάγνωση/ Εγγραφή

Πίνακας 1.1 Το σύστημα αρχείων στα τρία βασικά λειτουργικά συστήματα

Τα ήδη των αρχείων (**Ευσταθόπουλος**, 2001) που συναντά κανείς σε ένα σύστημα αρχείων είναι κυρίως πέντε:

1. **Κοινά αρχεία (regular files)**

   Τα κοινά αρχεία αποτελούν την απλούστερη μορφή αρχείου και σχεδόν πάντα είναι τα περισσότερα. Στα αρχεία αυτά γίνεται καθεαυτή η αποθήκευση της πληροφορίας και των δεδομένων κάθε είδους. Τα κοινά αρχεία είναι άμεσα προσπελάσιμα (ανάγνωση / εγγραφή) από τον χρήστη, αρκεί αυτός να διαθέτει τα κατάλληλα προνόμια. Οι περισσότερες κλήσεις συστήματος για συστήματα αρχείων μπορούν να κληθούν για αρχείου του τύπου αυτού.

2. **Αρχεία καταλόγων (directories)**

   Τα αρχεία καταλόγων (ή απλά κατάλογοι) βοηθούν στην οργάνωση των υπολοίπων αρχείων σε καταλόγους προσδίδοντας στο σύστημα αρχείων ιεραρχική δομή. Αν και οι κατάλογοι είναι αρχεία εμφανή στον χρήστη, δεν μπορεί κανείς να τα διαβάσει ή να τα τροποποιήσει με τρόπο παρόμοιο με τα απλά αρχεία. Η προσπέλαση των δεδομένων ενός καταλόγου είναι περιορισμένη και γίνεται μόνο μέσω των σχετικών κλήσεων συστήματος.

3. **Συμβολικοί σύνδεσμοι (symbolic links)**

   Οι συμβολικοί σύνδεσμοι είναι ειδικά αρχεία τα οποία χρησιμοποιούνται για λόγους εξοικονόμησης αποθηκευτικού χώρου και για άλλες διαχειριστικές σκοπιμότητες. Πρόκειται ουσιαστικά για

έναν τρόπο ονομασίας του ίδιου αρχείου με περισσότερους από έναν τρόπους. Η δημιουργία του συμβολικού συνδέσμου καταλαμβάνει ελάχιστο αποθηκευτικό χώρο, ενώ στη συνέχεια ο σύνδεσμος είναι εμφανής και προσβάσιμος στον χρήστη. Τους συμβολικούς συνδέσμους εισήγαγε το UNIX στο σύστημα αρχείων του και σήμερα υποστηρίζονται από όλα τα συστήματα αρχείων. που ακολουθούν τη φιλοσοφία αυτή. Αντίθετα, κάποια που βασίζονται σε άλλες σχεδιαστικές τεχνικές δεν υποστηρίζουν την έννοια του συμβολικού συνδέσμου (π.χ. vfat, NTFS)

4. **Αρχεία συσκευών (device files)**

Τα αρχεία συσκευών αποτελούν καινοτομία που εισήγαγε το λειτουργικό σύστημα UNIX στην αντιμετώπιση των συσκευών και των περιφερειακών. Το μοντέλο αντιμετώπισης των συσκευών σαν ειδικού τύπου αρχεία του Σ.Α. ακολουθείται από όλα τα λειτουργικά συστήματα που βασίζονται στη σχεδίαση του UNIX μέχρι και σήμερα. Κάθε συσκευή αναπαρίσταται σαν ένα ειδικό αρχείο μέσω του οποίου υλοποιείται η πρόσβαση τόσο στην ίδια τη συσκευή όσο και στα πιθανά περιεχόμενα της. Τα αρχεία αυτά είναι ορατά και προσβάσιμα στον χρήστη (εφόσον διαθέτει τα κατάλληλα προνόμια) και μπορούν να εφαρμοστούν σε αυτά οι περισσότερες κλήσεις συστήματος, με ιδιαίτερη όμως σημασία. Ακόμα, υπάρχουν και κάποιες ειδικές κλήσεις που αφορούν τη διαχείριση των συσκευών που αναπαριστούν τα αρχεία αυτά και οι οποίες εφαρμόζονται μόνο σε αρχεία του τύπου αυτού (π.χ. *iOctl).* Σημειώνεται ακόμη ότι, ενώ το άνοιγμα ενός υπαρκτού αρχείου συσκευής, η ανάγνωση και η εγγραφή του μπορούν να γίνουν όπως ακριβώς στα κοινά αρχεία, η δημιουργία των ειδικών αυτών αρχείων γίνεται μόνο μέσω της αρμόδιας κλήσης συστήματος *(mknod).*

5. **Αρχεία σωλήνωσης (pipes)**

Τα αρχεία σωλήνωσης αποτελούν ειδικού τύπου αρχεία που εξυπηρετούν λειτουργίες I/0. Είναι ορατά και προσπελάσιμα από τον χρήστη ενώ οι δυνατότητες εφαρμογής των κλήσεων για τα κοινά αρχεία σε αυτά είναι σχετικά περιορισμένες. Η δημιουργία των αρχείων σωλήνωσης γίνεται μέσω της αρμόδιας κλήσης *(pipe)*

1.5.3 Γενική ανάλυση της διεπαφής χρήστη ή κέλυφος (UI, GUI, Shell)

Όλα τα σύγχρονα λειτουργικά συστήματα έχουν γραφικό περιβάλλον (GUI) που τα κάνουν πολύ φιλικά προς το χρήστη. Το ίδιο ισχύει και στις διανομές του Linux και του Unix, όπου έχουμε ένα σύνολο από desktop environments από όπου μπορούμε να διαλέξουμε πιο θα χρησιμοποιήσουμε. Πολλές φορές θα πρέπει να χρησιμοποιήσουμε όμως ένα τερματικό για να κάνουμε κάποιες λειτουργίες, κυρίως

όταν πρέπει να αλλάξουμε αρχεία που πρόσβαση έχει μόνο ο διαχειριστής του συστήματος. Αυτό γίνεται ακόμα πιο έντονο στις εκδόσεις server όπου πολλές διανομές δεν έχουν εγκατεστημένο κάποιο desktop environment, κάτι που γίνεται για εξοικονόμηση πόρων του συστήματος.

Το shell είναι η διεπαφή (interface) επικοινωνίας του χρήστη με τον πυρήνα του λειτουργικού συστήματος. Το κέλυφος δέχεται εντολές τις οποίες μεταφράζει και στην συνέχεια εκτελεί. Τα βασικά είδη διασύνδεσης τα οποία έχουν επικρατήσει στα περισσότερα λειτουργικά συστήματα είναι η Γραφική Διεπαφή Χρήστη (Graphical User Interfac ή GUI), η Γραμμή Εντολών Κελύφους (Shell Command Line – ή SCL) η ή Διεπαφή Γραμμής Εντολών (Comand Line Interface ή CLI).

- ✓ **Γραμμή Εντολών Κελύφους (Shell Command Line) ή Διεπαφή Γραμμής Εντολών (Comand Line Interface ή CLI).**

Η Γραμμή Εντολών Κελύφους ή Διεπαφή Γραμμής Εντολών είναι από τις παλαιότερες (1960) διεπαφές χρήστη που συνεχίζουν να συναντώνται σε πολλά λειτουργικά είτε ως βασικός, είτε ως επιπλέον τρόπος αλληλεπίδρασης. Είναι αμεσότερο, αποτελεσματικότερο και καταναλώνει λιγότερους πόρους από το σύστημα. Δεν είναι όμως εύκολο στην χρήση για έναν απλό και μη εξειδικευμένο χρήστη καθώς απαιτεί καλή γνώση των εντολών, των παραμέτρων καθώς και της σύνταξης της γλώσσας που χρησιμοποιείται (**Τσιόκανος**, 2014)
Από το 1960 και μετά, η αλληλεπίδραση του χρήστη με τους υπολογιστές γινόταν κατά κύριο λόγο μέσω της διεπαφής γραμμής εντολών. Στην δεκαετία του 1970 άρχισε η έρευνα για την ανάπτυξη γραφικού περιβάλλοντος χρήστη (GUI) όπου όλες οι λειτουργίες θα γινόταν μέσω γραφικού περιβάλλοντος σε αντίθεση με την διεπαφή γραμμής εντολών που χρησιμοποιείται κείμενο. Μετά από αυτό το GUI είναι ο πιο συνηθισμένος τρόπος επικοινωνίας με τον υπολογιστή. Παρόλα αυτά η γραμμή εντολών δεν μπορεί να αντικατασταθεί πλήρως από ένα γραφικό περιβάλλον χρήστη. Ορισμένες εξειδικευμένες εργασίες μπορούν να γίνουν μόνο μέσω του CLI και των διαφόρων shell. (**A Brief History of User Interfaces**, 2016)

- ✓ **Γραφική Διεπαφή Χρήστη (Graphical User Inter face – GUI)**

Η Γραφική Διεπαφή Χρήστη, η GUI είναι ένα σύνολο γραφικών στοιχείων τα οποία δημιουργούν ένα γραφικό περιβάλλον οικείο και φιλικό προς τον χρήστη (**Τσιόκανος**, 2014) ώστε να επιτευχθεί η όσο το δυνατόν καλύτερη αλληλεπίδραση με το σύστημα. Το κύριο πλεονέκτημα της είναι ή ευκολία και απλότητα που παρέχει στον χρήστη.

Παραδείγματα GUI τα οποία είναι οικεία στους περισσότερους ανθρώπους σήμερα είναι τα περιβάλλοντα για σταθερούς και φορητούς υπολογιστές, όπως τα Microsoft Windows, το Mac OS X, και το X Window System, καθώς και αυτά για συσκευές χειρός ("smartphone") όπως το Symbian, το BlackBerry OS, το Android και το iOS της Apple. Η Apple, η IBM και η Microsoft χρησιμοποίησαν πολλές ιδέες της Xerox για να δημιουργήσουν προϊόντα, και οι προδιαγραφές για την κοινή πρόσβαση χρήστη (Common User Access) της IBM αποτέλεσαν τη βάση των περιβαλλόντων χρήστη που εντοπίζονται στα Microsoft Windows, στο IBM OS/2 κλπ. Αυτές οι ιδέες

εξελίχθηκαν για να δημιουργήσουν το περιβάλλον που εντοπίζεται στις τωρινές εκδοχές των Microsoft Windows, όπως επίσης και στα Mac OS X και σε ποικίλα περιβάλλοντα επιφάνειας εργασιών (desktop environments) για λειτουργικά συστήματα τύπου Unix, όπως το Linux. Έτσι τα πιο σύγχρονα GUI παρουσιάζουν κοινά χαρακτηριστικά σε μεγάλο βαθμό.

Μια σειρά στοιχείων που αυτά ανταποκρίνονται σε μια οπτική γλώσσα (visual language) εξελίχθηκαν για να αντιπροσωπεύουν πληροφορίες που είναι αποθηκευμένες στους υπολογιστές. Ο πιο κοινός συνδυασμός αυτών των στοιχείων είναι το WIMP (Window (παράθυρο), Image (εικόνα), Menu (μενού), Pointing Device (συσκευή κατάδειξης)), ειδικά σε προσωπικούς υπολογιστές. Το WIMP χρησιμοποιεί μια φυσική συσκευή εισόδου για να ελέγξει την θέση του κέρσορα (cursor) και παρουσιάζει πληροφορίες οι οποίες οργανώνονται σε παράθυρα και αντιπροσωπεύονται με εικονίδια. Οι διαθέσιμες εντολές καταρτίζονται μαζί σε μενού, και ενέργειες εκτελούνται μέσω της συσκευής ενδείξεων. Ένας διαχειριστής παραθύρων (window manager) διευκολύνει τις αλληλεπιδράσεις ανάμεσα στα παράθυρα, τις εφαρμογές, και το σύστημα παραθύρων (windowing system). Το σύστημα παραθύρων χειρίζεται συσκευές υλικού όπως οι συσκευές ενδείξεων, το υλικό γραφικών αλλά και την θέση του κέρσορα. Στους προσωπικούς υπολογιστές όλα αυτά τα στοιχειά συνδυάζονται μέσω μεταφοράς στην επιφάνεια εργασίας, για να παράγουν μια προσομοίωση που ονομάζεται περιβάλλον επιφάνειας εργασίας στο οποίο η οθόνη παριστάνει μια επιφάνεια εργασίας πάνω στην οποία έγγραφα και φάκελοι μπορούν να τοποθετηθούν. Οι διαχειριστές παραθύρων και άλλου λογισμικού συνδυάζονται για να προσομοιώσουν το περιβάλλον επιφάνειας εργασίας με ποικίλα ρεαλιστικά στοιχεία. (**A Brief History of User Interfaces**, 2016)

Βασικά στοιχεία του GUI (**Κατσούλας** κ.α.) είναι τα εξής:

- ✓ Η χρήση περιβάλλοντος γραφικών και όχι απλής γραμμής κειμένου.
- ✓ Η χρήση «παραθύρων» που είναι ορθογώνιες περιοχές στην οθόνη εντός των οποίων μπορούν να εκτελούνται εφαρμογές. Τα παράθυρα αυτά έχουν έναν καθορισμένο τρόπο εμφάνισης και χειρισμού (τίτλος παραθύρου, χρώματα πλαισίου, μεγιστοποίηση, ελαχιστοποίηση, κλείσιμο).
- ✓ Ύπαρξη μιας «επιφάνειας εργασίας» στην οποία μπορούν να υπάρχουν τα παράθυρα των εκτελούμενων εφαρμογών καθώς και εικονίδια που αντιστοιχούν σε εφαρμογές, αρχεία και φακέλους.
- ✓ Πλήρης έλεγχος μέσω συσκευών κατάδειξης (π.χ ποντίκι, light pen, touchpad, οθόνη αφής). Ο χρήστης μπορεί να εκκινήσει και να τερματίσει εφαρμογές, να αλλάξει θέση και μέγεθος στα παράθυρα και γενικά να προκαλεί «γεγονότα» (π.χ πάτημα πλήκτρου ποντικιού, τοποθέτηση δείκτη ποντικιού σε μια περιοχή, κ.ά) τα οποία διαχειρίζονται από το λειτουργικό σύστημα και οδηγούνται προς τις αντίστοιχες εφαρμογές για έλεγχο και ανταπόκριση.
- ✓ Δυνατότητα ύπαρξης περιοχής όπου υπάρχει ένα «μενού» των εφαρμογών.
- ✓ Δυνατότητα ύπαρξης μιας περιοχής όπου εμφανίζονται (σε μορφή εικονιδίων) οι εφαρμογές που εκτελούνται και διάφορες άλλες πληροφορίες (π.χ ώρα)

Κεφάλαιο 2

2.1 Σύντομη αναφορά στην ιστορία του UNIX

Το UNIX αναπτύχθηκε αρχικά στα εργαστήρια της Bell Labs και άρχισε να λειτουργεί σε έναν PDP-7 το 1970. Μερικοί από τους ανθρώπους που συμμετείχαν στα εργαστήρια της Bell Labs είχαν επίσης συμμετάσχει στις εργασίες time-sharing που έγιναν στο MIT για το πρόγραμμα MAC. Το σχέδιο αυτό οδήγησε στην ανάπτυξη του πρώτου CTSS[1] και στη συνέχεια του Multics[2]. Η αρχική έκδοση του UNIX και οι προγραμματιστές του επηρεάστηκαν περισσότερο από το CTSS. Παρ 'όλα αυτά, το UNIX ενσωματώνει πολλές ιδέες και από το Multics.

Οι εργασίες σχετικά με το UNIX στα εργαστήρια της Bell Labs, και αργότερα αλλού, παρήγαγαν μια σειρά από εκδόσεις του UNIX. Η πρώτη αξιοσημείωτη έκδοση και διαδικασία ήταν η εξαγωγή και φόρτωση του συστήματος UNIX από τον PDP-7 στον PDP-11. Αυτή ήταν η πρώτη νύξη, για το ότι το Unix θα ήταν ένα λειτουργικό σύστημα για όλους τους υπολογιστές. Η επόμενη σημαντική διαδικασία και επίτευξη ήταν η συγγραφή του UNIX στη γλώσσα προγραμματισμού C. Αυτή ήταν μια ανήκουστη στρατηγική εκείνη την εποχή. Γενικά υπήρχε η άποψη ότι κάτι τόσο περίπλοκο όπως ένα λειτουργικό σύστημα, το οποίο πρέπει να ασχολείται με το διαμοιρασμό και την σωστή κατανομή του χρόνου των κρίσιμων γεγονότων, έπρεπε να γραφτεί αποκλειστικά σε συμβολική γλώσσα. Η εφαρμογή της γλώσσας C στο UNIX κατέδειξε τα πλεονεκτήματα της χρήσης μιας γλώσσας υψηλού επιπέδου για όλα τα υποσύνολα και τις συνιστώσες, αν όχι για όλο των κώδικα του συστήματος. Σήμερα, σχεδόν όλες οι εφαρμογές του UNIX είναι γραμμένες στη C.

Αυτές οι πρώιμες εκδόσεις του UNIX ήταν δημοφιλείς στο εσωτερικό των εργαστηρίων της Bell Labs. Το 1974, το σύστημα UNIX περιγράφηκε σε ένα τεχνικό περιοδικό για πρώτη φορά. Αυτό προκάλεσε μεγάλο ενδιαφέρον για το σύστημα. Άδειες για το UNIX δόθηκαν σε εμπορικά ιδρύματα, καθώς και σε πανεπιστήμια. Η πρώτη ευρέως διαθέσιμη έκδοση έξω από την Bell Labs ήταν η έκδοση 6, το 1976. Η συνέχεια της ήταν η έκδοση 7, που κυκλοφόρησε το 1978, η οποία είναι ο πρόγονος των πιο σύγχρονων συστημάτων UNIX. Το πιο σημαντικό από τα συστήματα και τα τμήματα του λειτουργικού που δεν αναπτύχθηκαν στη AT&T έγινε στο Πανεπιστήμιο της Καλιφόρνια στο Μπέρκλεϊ, και ονομάστηκε UNIX BSD (Berkeley Software Distribution), και τρέχει για πρώτη φορά στο PDP και στη συνέχεια σε μηχανές VAX. Η AT & T συνέχισε να αναπτύσσει και να τελειοποιεί το σύστημα. Από το 1982, η Bell Labs είχε συνδυάσει αρκετές. παραλλαγές του UNIX σε ένα ενιαίο σύστημα, το οποίο κυκλοφόρησε στο εμπόριο με όνομα Unix III. Ένας αριθμός χαρακτηριστικών προστίθενται αργότερα στο λειτουργικό σύστημα για να παραχθεί το Unix V.

[1]The Compatible Time Sharing System (1961–1973), Fiftieth Anniversary Commemorative Overview, David Walden, Tom Van Vleck, IEEE Computer Society, 2001 και Reminiscences on the history of time sharing, John McCarthy, Stanford University, 1983

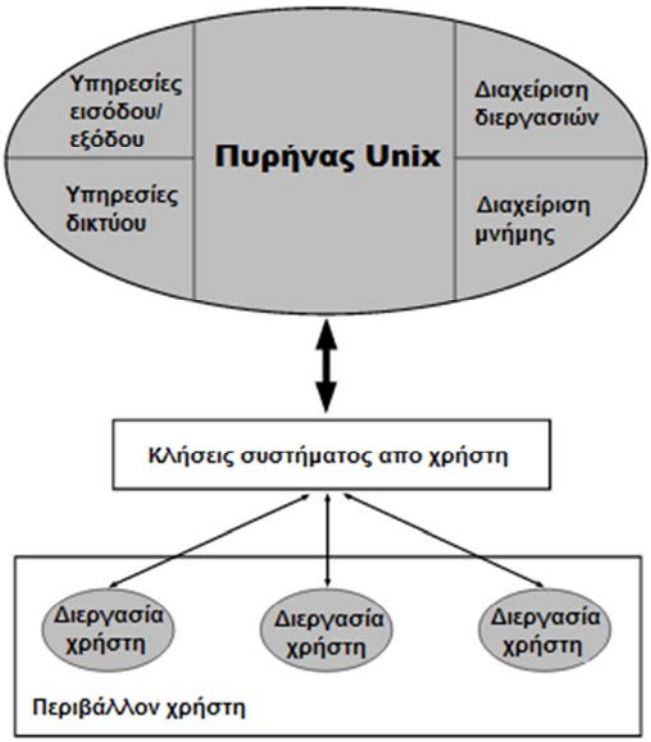
[2]The multiplexed information and computing service, Massachusetts Institute of Technology (MIT), 1973

2.2 Περιγραφή του συστήματος

Παρά την απλότητα του λειτουργικού συστήματος το UNIX έχει εξελιχθεί σε ένα πολύ πολύπλοκο, ευέλικτο και επεκτάσιμο λειτουργικό σύστημα. Η πολυπλοκότητα αυτή οφείλεται στο γεγονός ότι ο πυρήνας διαχειρίζεται μια μεγάλη ποικιλία υπηρεσιών (***Inside the unix kernel***, 2000, p.1), συμπεριλαμβανομένων του δικτυακού συστήματος αρχείων (NFS), την Είσοδο/Εξόδο -(I/O), τις υποδοχές (sockets), καθώς και τις διεργασίες και τη διαχείριση μνήμης. Το UNIX είναι ένα εξαιρετικά παραμετροποιήσιμο σύστημα και υποστηρίζει πολλούς διαφορετικούς τύπους διαμόρφωσης.

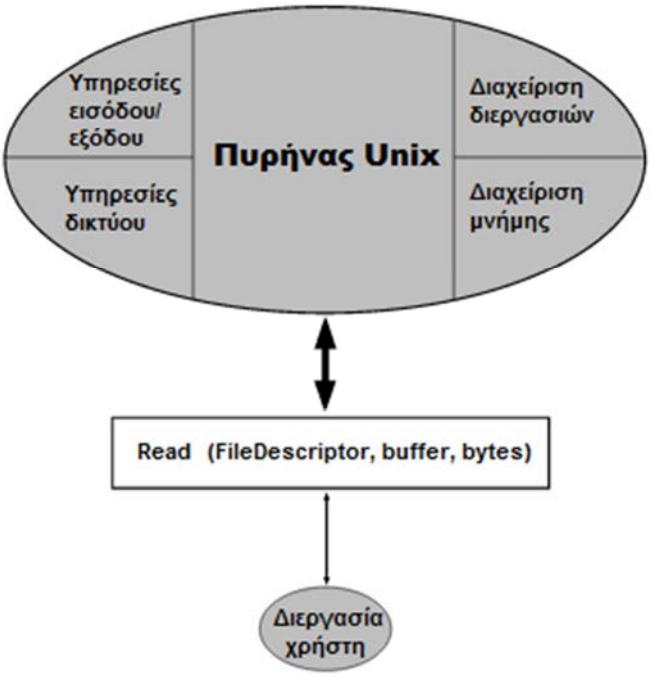
2.2.1 Το εσωτερικό του πυρήνα

Ο πυρήνας του UNIX είναι ο κεντρικός μηχανισμός του λειτουργικού συστήματος (***Inside the unix kernel***, 2000, p.1-2). Παρέχει διασύνδεση με τις συσκευές υλικού, καθώς και με την ΚΜΕ, τη μνήμη, και τη διαχείριση της Ε/Ε. Ο πυρήνας διαχειρίζεται τα αιτήματα από τους χρήστες μέσω των κλήσεων συστήματος και με αυτόν τον τρόπο μεταβαίνει από την λειτουργία χρήστη/χώρος χρήστη στην λειτουργία πυρήνα/χώρος πυρήνα (Σχήμα 2.1).



Σχήμα 2.1 Ο πυρήνας του Unix και οι κλήσεις συστήματος (Πηγή: pearsonhighered.com)

Κάθε φορά που μια διεργασία από τον χρήστη κάνει μια κλήση συστήματος, όπως π.χ. `read()`, `fork()`, `exec()`,`open()`, και ούτω καθεξής, η διεργασία χρήστη αντιμετωπίζει ένα context switch. Το context switch είναι ένας μηχανισμός με τον οποίο μια διεργασία μεταβαίνει από μία κατάσταση σε μία άλλη. Η διεργασία είτε μπορεί να ανασταλεί μέχρι να ολοκληρωθεί η κλήση συστήματος (blocking), ή η διαδικασία μπορεί να συνεχιστεί και αργότερα να ενημερωθούν για την ολοκλήρωση της κλήσης συστήματος μέσω ενός σήματος (non blocking). Το σχήμα 2.2 δείχνει ένα παράδειγμα ενός context switch



Σχήμα 2.2 Κλήση συστήματος από χρήστη για εντολή διάβασε (Πηγή: pearsonhighered.com)

Το σχήμα 2.2 δείχνει μια διεργασία χρήστη η οποία κάνει χρήση της κλήσης συστήματος. Η κλήση συστήματος `read-()` λαμβάνει ως παραμέτρους έναν περιγραφέα αρχείου (file descriptor), το buffer στον οποίο θα τοποθετηθούν τα δεδομένα της ανάγνωσης, και το πλήθος των bytes που θα διαβαστούν. Η κλήση συστήματος `read-()` αναγκάζει τη διεργασία χρήστη να μπλοκαριστεί έως ότου ολοκληρωθεί ή επέλθει το όριο χρόνου της κλήσης συστήματος `read-()`.
Ο πυρήνας του UNIX παρέχει υπηρεσίες πρόσβασης σε διάφορους πόρους του συστήματος, συμπεριλαμβανομένων της Ε/Ε, του διαχειριστή μνήμης, του διαχειριστή των διεργασιών, και των υπηρεσιών δικτύου. Μια εφαρμογή ή μία διεργασία χρήστη μπορεί να έχει πρόσβαση στους πόρους και τις υπηρεσίες του συστήματος μέσω των κλήσεων συστήματος. Η απόδοση της εφαρμογής εξαρτάται από τον τύπο των κλήσεων συστήματος που θα χρησιμοποιηθεί και από τον αριθμό των κλήσεων που θα κάνει η κάθε εφαρμογή στο σύστημα. Οι πόροι του πυρήνα είναι και θα πρέπει να θεωρούνται ως ένα πολύτιμο και περιορισμένο σύνολο. Με σκοπό την μεγιστοποίηση της απόδοσης του συστήματος, η εφαρμογή θα πρέπει να ελαχιστοποιεί την ποσότητα των κλήσεων, μειώνοντας έτσι την κατάληψη χώρου του

πυρήνα (Kernel space) και διατηρώντας τις διεργασίες των χρηστών στον χώρο χρήστη (User space) για το μεγαλύτερο μέρος του χρόνου. Ωστόσο, ορισμένες εφαρμογές, όπως ένα σύστημα διαχείρισης βάσεων δεδομένων εξαρτάται σε μεγάλο βαθμό από την Ε/Ε, και σε αυτή την περίπτωση δεν μπορεί να αποφεύγονται οι κλήσεις. Στόχος αυτής της κατάστασης είναι να ρυθμιστεί ο πυρήνας του UNIX και το σύστημα έτσι ώστε να ανταποκρίνονται όσο το δυνατόν πιο γρήγορα και αποτελεσματικά οι πόροι του πυρήνα και να μπορούν να απελευθερωθούν γρήγορα για την εξυπηρέτηση άλλων αιτημάτων. Με άλλα λόγια, ο στόχος των αιτήσεων και εφαρμογών θα πρέπει να είναι η ελαχιστοποίηση της ποσότητας του χρόνου που δαπανάται στον χώρο του πυρήνα, ενώ παράλληλα να μεγιστοποιείται η απόδοση στο χώρο του χρήστη.

Ένα από τα σημαντικότερα ζητήματα και κεντρικής σημασίας για το λειτουργικό σύστημα του UNIX είναι η κλήση συστήματος `fork`[1]`-().` Η κλήση αυτή δημιουργεί μια άλλη διεργασία που είναι γνωστή ως *child process*, ενώ η διεργασία που εξέδωσε την κλήση συστήματος fork () γίνεται η γονική διεργασία (*parent process*). Η κλήση συστήματος `fork-()` αναπαράγει και διπλασιάζει τη συνολική δομή της διεργασίας και το χώρο διευθύνσεων της γονικής διεργασίας. Η *child process* κληρονομεί τα χαρακτηριστικά της πρωταρχικής κλάσης και το περιβάλλον της γονικής διεργασίας (*parent process*). Η *child process* κληρονομεί επίσης τους ανοιχτούς file descriptors από τη γονική διεργασία. Η κλήση συστήματος `fork-().` επιτρέπει σε μία διεργασία να δημιουργεί μια άλλη διεργασία, είτε συγχρονισμένα από την αναμονή για τη *child process*, ή ασύγχρονα με τη συνέχιση της εκτέλεσης της γονικής διεργασίας. Η κλήση συστήματος `fork-()` δεν δέχεται ορίσματα και επιστρέφει έναν ακέραιο. Η τιμή επιστροφής μπορεί να αντιπροσωπεύει μία από τις τρεις παρακάτω διακριτές τιμές:

- ✓ 0, που σημαίνει ότι η κλήση `fork-()` δημιούργησε με επιτυχία μια *child process*, το 0 αναφέρεται στη *child process*, μέσω της διεργασίας της γονικής
- ✓ -1, που σημαίνει ότι το σύστημα δεν ήταν σε θέση να δημιουργήσει μια άλλη διεργασία. Σε αυτήν την περίπτωση, μπορούμε να χρησιμοποιήσουμε την κλήση `perror-()` για την ανάγνωση του ακριβούς μηνύματος λάθους που προκάλεσε την αποτυχία της κλήσης `fork-()` ή
- ✓ ένας προεπιλεγμένος θετικός ακέραιος αριθμός μεγαλύτερος του 0 που επιστρέφεται στη γονική διεργασία και που αντιπροσωπεύει το αναγνωριστικό (ID) της *child process*. Η γονική διεργασία μπορεί στη συνέχεια να περιμένει για τη *child process*, εμποδίζοντας έτσι μέχρι τον τερματισμό της *child process*, ή η γονική διεργασία μπορεί να συνεχίσει την εκτέλεση της, χωρίς να περιμένει τη *child process*. Αυτή θα ήταν μια ασύγχρονη περίπτωση, και η γονική διεργασία μπορεί να ενημερωθεί όταν η *child process* τερματιστεί. Η γονική διεργασία θα πρέπει να καθιερώσει ένα τμήμα κώδικα ο οποίος να διαχειρίζεται τα σήματα έτσι ώστε να παγιδεύονται και να ερμηνεύονται τα σήματα από τη *child process*

[1]Η `fork` είναι μια ενέργεια με την οποία μια διεργασία δημιουργεί ένα αντίγραφο του εαυτού της. Είναι συνήθως μια κλήση συστήματος, που διενεργείτε στον πυρήνα. Η `fork` είναι η κύρια (και ιστορικά μοναδική) μέθοδος δημιουργίας διεργασίας για τα Unix-like λειτουργικά συστήματα.

Η κλήση συστήματος `fork-()` δεν χρησιμοποιείται μόνο στο πλαίσιο των προγραμμάτων/εφαρμογών για τη δημιουργία υπό διεργασιών, αλλά και μέσα στον ίδιο τον πυρήνα για τη δημιουργία επιμέρους διεργασιών. Για παράδειγμα, ας εξετάσουμε την έξοδο των αποτελεσμάτων στον Πίνακα 2.1 που μας δίνει η εντολή `ps-ef.`

UID	PID	PPID	TIME CMD
root	0	0	0:01 sched
root	1	0	0:01 /etc/init -
root	2	0	0:00 pageout
root	3	0	3:27 fsflush
root	131	1	0:01 /usr/sbin/inetd -s
root	289	1	0:00 /usr/lib/saf/sac -t 300
root	112	1	0:02 /usr/sbin/rpcbind
root	186	1	0:00 /usr/lib/lpsched
root	104	1	0:12 /usr/sbin/in.routed -q
root	114	1	0:01 /usr/sbin/keyserv
root	122	1	0:00 /usr/sbin/kerbd
root	120	1	0:00 /usr/sbin/nis_cachemgr
root	134	1	0:00 /usr/lib/nfs/statd
root	136	1	0:00 /usr/lib/nfs/lockd
root	150	1	0:00 /usr/lib/autofs/automountd
root	195	1	0:00 /usr/lib/sendmail -bd -q1h
root	159	1	0:00 /usr/sbin/syslogd
root	176	1	0:07 /usr/sbin/nscd
root	169	1	0:13 /usr/sbin/cron

Πίνακας 2.1 Αποτελέσματα της εντολής ps –ef (Πηγή: pearsonhighered.com)

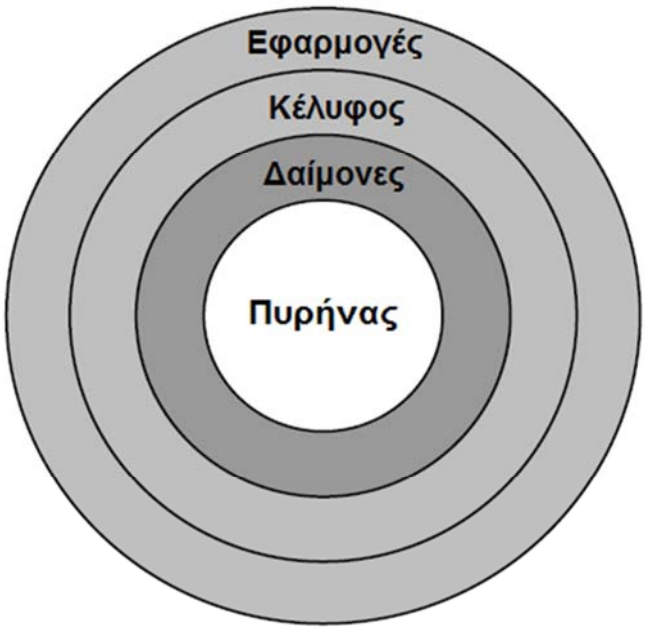
Στον πίνακα 2.1, η διεργασία `sched` έχει το αναγνωριστικό διεργασίας (PID) 0 και το αναγνωριστικό της γονικής διεργασίας (PPID) 0, που σημαίνει ότι η `sched` διαδικασία είναι η αρχική βασική διαδικασία. Η διαδικασία init έχει το αναγνωριστικό διεργασίας ID 1 και η γονική διεργασία το ID 0, πράγμα που σημαίνει ότι η διεργασία init κληρονομήθηκε από τη `sched` διεργασία. Η `sched` διεργασία είναι η διεργασία προγραμματιστής (scheduler process) που είναι υπεύθυνη για διεργασίες προγραμματισμού στην ουρά (run queue). Ο inetd δαίμονας έχει ένα αναγνωριστικό διαδικασίας 131 και η γονική διεργασία 1, που σημαίνει ότι η υπηρεσία inetd κληρονομήθηκε από τη διεργασία init. Η init είναι η διεργασία αρχικοποίησης συστήματος που είναι υπεύθυνη για την ωοτοκία και την προετοιμασία των διαδικασιών με ορισμένες προεπιλογές.
Ως εκ τούτου, είναι προφανές ότι η κλήση συστήματος `fork-()` παίζει σημαντικό ρόλο στο UNIX, και χρησιμοποιείται συχνά τόσο στις εφαρμογές χρήστη όσο και στις εφαρμογές του συστήματος. Ας εξετάσουμε το ακόλουθο τμήμα προγράμματος που συνοψίζει την κλήση της `fork-()` στο σύστημα και τη χρήση της.

```
#define FORK_ERROR -1
#include <stdio.h>
#include <unistd.h>
main ()
{
      int lpid;
      lpid=fork(); /* fork() issued, return code is checked below */
      switch(lpid) /* for success or failure. */
      {
            case 0: /* Child Process section - fork() succeeded, call
                  execl(). */
               execl ("sample2",arg*0,arg*1,....,arg*n, NULL);

            case -1: /* Unable to create process */
                  perror ("Unable to create process");
                  exit (FORK_ERROR);

            default: /* Return to Parent Process */
                  /* Either continue or wait on child process. */
            /* Value returned here is process id of child process. */
      }
}
```

2.2.2 Αρχιτεκτονική του Πυρήνα

Το λειτουργικό σύστημα του UNIX αποτελείται από πολλά διαφορετικά στρώματα (layers) που διαχειρίζονται διαφορετικούς πόρους και υπηρεσίες. Η βασική αρχιτεκτονική του UNIX (OS) αποτελείται από δύο (***Inside the unix kernel***, 2000). κύρια στρώματα:

- ✓ το σύστημα ή στρώμα πυρήνα
- ✓ και το στρώμα χρήστη.



Σχήμα 2.3 Τα διάφορα στρώματα του UNIX (Πηγή: pearsonhighered.com)

Το στρώμα (layer) του πυρήνα είναι το κέντρο του λειτουργικού συστήματος και παρέχει υπηρεσίες όπως η επεξεργασία και η διαχείριση της μνήμης, οι υπηρεσίες Ε/Ε, τα προγράμματα οδήγησης συσκευών, ο χρονοπρογραμματισμός των εργασιών, οι ρουτίνες υλικού σε χαμηλό επίπεδο. Το στρώμα πυρήνα αποτελείται από πολλές διαφορετικές εσωτερικές ρουτίνες που επεξεργάζονται τα αιτήματα των χρηστών. Η πρόσβαση στον πυρήνα είναι έμμεση, μέσω των κλήσεων συστήματος. Δεν είναι δυνατό να καλέσουμε τις εσωτερικές ρουτίνες του πυρήνα άμεσα. Οι κλήσεις συστήματος παρέχουν ένα μηχανισμό με τον οποίο μια διαδικασία χρήστη μέσω του συστήματος κλήσεων μπορεί να ζητήσει έναν πόρο πυρήνα ή μία υπηρεσία προς διάθεση. Ο ίδιος ο πυρήνας είναι σχετικά μικρός σε σχέση με τα μεγέθη δίσκων και μνημών αποθήκευσης που υπάρχουν. Σε συστήματα Sequent, ο πυρήνας βρίσκεται στον ριζικό κατάλογο και αποτελείται από ένα ενιαίο αρχείο το οποίο ονομάζεται unix (`/unix`). Το μέγεθος του πυρήνα του UNIX σε συστήματα Sequent ποικίλλει ανάλογα με τη διαμόρφωση του συστήματος και τον αριθμό των οδηγών των συσκευών που υπάρχουν και έχουν εγκατασταθεί.
Ένα τυπικό μέγεθος μπορεί να κυμαίνεται από 1 MB έως 5 MB. Στο λειτουργικό σύστημα Solaris 7 (2,7), η διαδρομή του αρχείου για τον πυρήνα του UNIX είναι `/platform/<platform_type>/kernel/unix`. Το `< platform_type >` υποδεικνύει τον τύπο της αρχιτεκτονικής του συστήματος, όπως οι `sun4m`[1] ή `sun4u`[1]. Είναι σημαντικό να σημειωθεί ότι όταν το λειτουργικό σύστημα εγκατασταθεί, ο υπολογιστής/μηχάνημα θα πρέπει να διαθέτει αρκετό χώρο για το ριζικό σύστημα αρχείων. Αν δεν διαθέτει επαρκή χώρο και θελήσουμε να επαναδημιουργήσουμε τον πυρήνα (rebuild the kernel) μπορεί αυτό να μην είναι δυνατόν να γίνει, διότι η επανοικοδόμηση του πυρήνα απαιτεί μια προσωρινή περιοχή ανασυγκρότησης για να κρατήσει τόσο τον νέο πυρήνα όσο και τον προηγούμενο. Επιπλέον, μπορεί να μην είναι δυνατόν να εγκαταστήσουμε patches (ενημερωμένες εκδόσεις κώδικα – κάλυψη κενών ασφαλείας και ενημέρωση αυτών) στο λειτουργικό μας σύστημα εάν επαρκής ελεύθερος χώρος δεν υπάρχει στο ριζικό σύστημα αρχείων. Συνίσταται να υπάρχει πάντα τουλάχιστον 1GB ελεύθερου χώρου στο ριζικό σύστημα αρχείων. Αν και αυτός μπορεί να φαίνεται αρκετά μεγάλο χώρος, η ανοικοδόμηση (rebuild) του συστήματος αρχείων στο ριζικό για την αύξηση του χώρου, δεν είναι καθόλου εύκολο να γίνει εκ τον υστέρων. Είναι καλύτερο να έχουμε επιπλέον χώρο από το να τσιγκουνευτούμε τα gb.

Ο πυρήνας παρέχει επίσης υπηρεσίες όπως ο χειρισμός των σημάτων, συγχρονισμό, ενδοεπικοινωνία, υπηρεσίες του συστήματος αρχείων, υπηρεσίες δικτύου και την παρακολούθηση του υλικού. Κάθε φορά που ξεκινά μια διαδικασία, ο πυρήνας έχει να προετοιμάσει τη διαδικασία, και να εκχωρήσει στη διαδικασία κάποια προτεραιότητα, να κατανείμει τη μνήμη και τους πόρους για τη διαδικασία και το χρονοδιάγραμμα της διαδικασίας για να τρέξει. Όταν η διαδικασία τερματιστεί, ο πυρήνας απελευθερώνει οποιαδήποτε μνήμη ή / και τους πόρους που ήταν κατειλημμένοι από τη διαδικασία

[1]Το Sun-4 είναι μια σειρά από σταθμούς εργασίας και servers του Unix που παράγεται από την Sun Microsystems, η οποία ξεκίνησε το 1987.

2.2.3 Διαδικασίες και διεργασίες στο Unix

Το λειτουργικό σύστημα UNIX χρησιμοποιεί μια αρχιτεκτονική προτεραιότητας (***Inside the unix kernel***, 2000) για τις διεργασίες με βάση τον αλγόριθμο round-robin[1]. Οι διεργασίες αποστέλλονται στην ουρά για να τρέξουν με βάση την προτεραιότητα και τοποθετούνται στην ουρά αναμονής (sleep queue) είτε λόγω λήξης της μονάδας χρόνου ή λόγω αναμονής για ένα γεγονός ή πόρο. Οι διεργασίες μπορούν μερικές φορές να εμπλακούν σε αδιέξοδο, όταν αναμένουν κυκλικά κάποιον πόρο να γίνει διαθέσιμος ενώ ταυτόχρονα κρατούν δεσμευμένους κάποιους άλλους πόρους. Το λειτουργικό σύστημα πρέπει να διασφαλίζει την ανίχνευση/αποτροπή των αδιέξοδων για να μη δημιουργούνται αδιέξοδα, κάτι που οδηγεί σταδιακά σε σταμάτημα της λειτουργίας του ΥΣ.

Όταν ένας χρήστης δίνει την εντολή `myprogram` στο UNIX, ο πυρήνας τοποθετεί το αίτημα για το `myprogram` στην ουρά αποστολής με ένα συγκεκριμένο επίπεδο προτεραιότητας. Η προεπιλεγμένη προτεραιότητα για την συγκεκριμένη κατηγορία εργασίας ανατίθεται, υπό την προϋπόθεση ότι δεν έχει ζητηθεί ειδική προτεραιότητα είτε μέσω της εντολής `nice`[2] ή μέσω της `priocntl-()`[3]. Οι εσωτερικές ρουτίνες του πυρήνα σαρώνουν την ουρά αποστολής (dispatch queue) και την ουρά αναμονής (sleep queue) για να εξασφαλίσουν ότι οι διεργασίες θα έχουν διαθέσιμους πόρους από την CPU και ότι δεν θα υπερβούν το χρόνο που τους έχει δοθεί. Οι εσωτερικές ρουτίνες πυρήνα δεν θα πρέπει να συγχέονται με τις κλήσεις συστήματος. Οι εσωτερικές ρουτίνες πυρήνα είναι ρουτίνες που μπορούν να χρησιμοποιηθούν και να ζητηθούν μόνο από τον πυρήνα. Οι κλήσεις συστήματος είναι η σύνδεση του χρήστη με το σύστημα υπηρεσιών του πυρήνα. Για παράδειγμα οι `fork-()` και `read-()` είναι παραδείγματα των κλήσεων συστήματος από τον χρήστη, ενώ οι `getpage-()` και `putpage-()` είναι παραδείγματα εσωτερικών ρουτινών διαχείρισης μνήμης. Το σχήμα 2.4 απεικονίζει τον κύκλο προγραμματισμού των διεργασιών του UNIX.

Το σχήμα 2.4 δείχνει τόσο την ουρά αποστολής όσο και την ουρά αναμονής καθώς περιέχει διεργασίες. Η ουρά αποστολής περιλαμβάνει διεργασίες που σημειώνονται ως έτοιμες να τρέξουν. Μόλις μια διεργασία επισημανθεί ως εκτελέσιμη, η διεργασία θα ξεκινήσει να εκτελείται στη CPU, μόλις η CPU γίνει διαθέσιμη. Στο σχήμα 2.4, τρεις διεργασίες εμφανίζονται στην ουρά αποστολής όπου στη συνέχεια αρχίζουν να εκτελούνται στους τρεις διαθέσιμους επεξεργαστές.

---

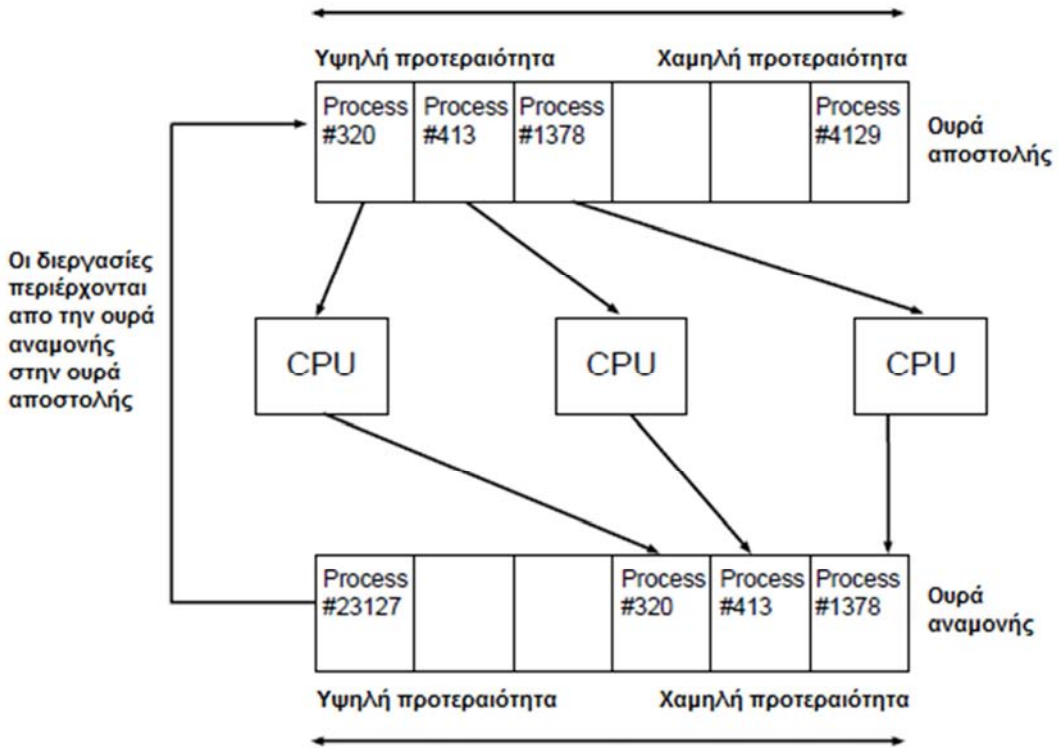
[1]Ο Round-robin (RR) είναι ένας αλγόριθμους που χρησιμοποιείται από προγραμματιστικές διεργασίες και διεργασίες δικτύου στην πληροφορική. Όπως ο όρος χρησιμοποιείται γενικά, οι χρονοθυρίδες εκχωρούνται σε κάθε διεργασία σε ίσες ποσότητες και σε κυκλική διάταξη, έτσι ώστε ο χειρισμός όλων των διεργασιών να γίνεται χωρίς προτεραιότητες (επίσης γνωστή και ως κυκλική εκτέλεση). Ο προγραμματισμός των διεργασιών με τον Round-robin είναι απλός και εύκολος στην εφαρμογή. Ο Round-robin μπορεί επίσης να εφαρμοστεί και σε άλλα προβλήματα προγραμματισμού, όπως ο προγραμματισμός πακέτων δεδομένων σε δίκτυα υπολογιστών.

[2]Η εντολή `nice` επιτρέπει στο χρήστη να αλλάξει την προτεραιότητα μιας συγκεκριμένης διεργασίας. Μόνο ο διαχειριστής (superuser) ή ο λογαριασμός root μπορεί να αυξήσει την προτεραιότητα μιας διεργασίας χρησιμοποιώντας την εντολή nice.

[3]Η `priocntl-()` παρέχει μια διεπαφή κλήσης συστήματος στο διαχειριστή προτεραιοτήτων των διεργασιών

Μόλις ο χρόνος που έχει αποδοθεί σε κάθε διεργασία (time-sharing job class) τερματιστεί, η διεργασία περιέρχεται στην ουρά αναμονής και αυτό έχει ως αποτέλεσμα οι άλλες διεργασίες οι οποίες ήταν στην ουρά αναμονής, τώρα να επισημανθούν ως έτοιμες για να τρέξουν.

Ο αλγόριθμος προγραμματισμού των διεργασιών στον πυρήνα του UNIX προσπαθεί να ισοκατανείμει το χρόνο της CPU μεταξύ των διεργασιών, προκειμένου να αποφευχθεί η αδυναμία της CPU να παρέχει πόρους στις διάφορες διεργασίες. Μόλις οι διεργασίες μεταφερθούν στην ουρά αναμονής, η διεργασία με την υψηλότερη προτεραιότητα θα προγραμματιστεί να τρέξει. Το ίδιο ισχύει και στην ουρά αποστολής. Ο πυρήνας μπορεί επίσης να διακόψει μία διεργασία (preempt processes), ούτως ώστε να μειώσει την προτεραιότητα των διεργασιών. Αυτή είναι μια τεχνική που χρησιμοποιείται από τον πυρήνα του UNIX για να σταματήσει μια διεργασία που τρέχει ή το σύνολο των διεργασιών και να εκκινήσει άλλες διεργασίες και να τις προσθέσει στο χρονοδιάγραμμα. Μια διεργασία μπορεί να διακόπτεται από τον πυρήνα, υπό πολλές διαφορετικές περιστάσεις, όπως όταν η διεργασία έχει υπερβεί το χρόνο που της έχει ανατεθεί, μία διεργασία με υψηλότερη προτεραιότητα πρέπει να χρονοπρογραμματιστεί για εκτέλεση, εάν προκύψει ένα συμβάν συστήματος ή μία διακοπή που πρέπει να εξυπηρετηθεί, ή η διαδικασία φαίνεται πως περιμένει για ένα πόρο που είναι απασχολημένος να τις διατεθεί. Ο πυρήνας διατηρεί πληροφορίες σχετικά με τις διεργασίες, συμπεριλαμβανομένης της τρέχουσας κατάστασης των διεργασιών. Χρησιμοποιώντας την εντολή `ps`, μπορούμε να δούμε την κατάσταση μιας συγκεκριμένης διαδικασίας ή τη λίστα των διεργασιών. Για παράδειγμα, ο πίνακας 2.2 δείχνει την έξοδο από την εντολή `ps`



Σχήμα 2.4 Κύκλος προγραμματισμού των διεργασιών στο Unix (Πηγή: pearsonhighered.com)

ST	UID	PID	PPID	TIME	CMD
S	root	0	0	0:01	Sched
S	root	1	0	0:01	/etc/init-
O	root	2	0	0:00	pageout
R	test	281	280	0:01	xterm
Z	test	282			<defunct>

Πίνακας 2.2 Αποτελέσματα της εντολής ps (Πηγή: pearsonhighered.com)

Η κατάσταση των διεργασιών μπορεί να χαρακτηριστεί με βάση τα παρακάτω:

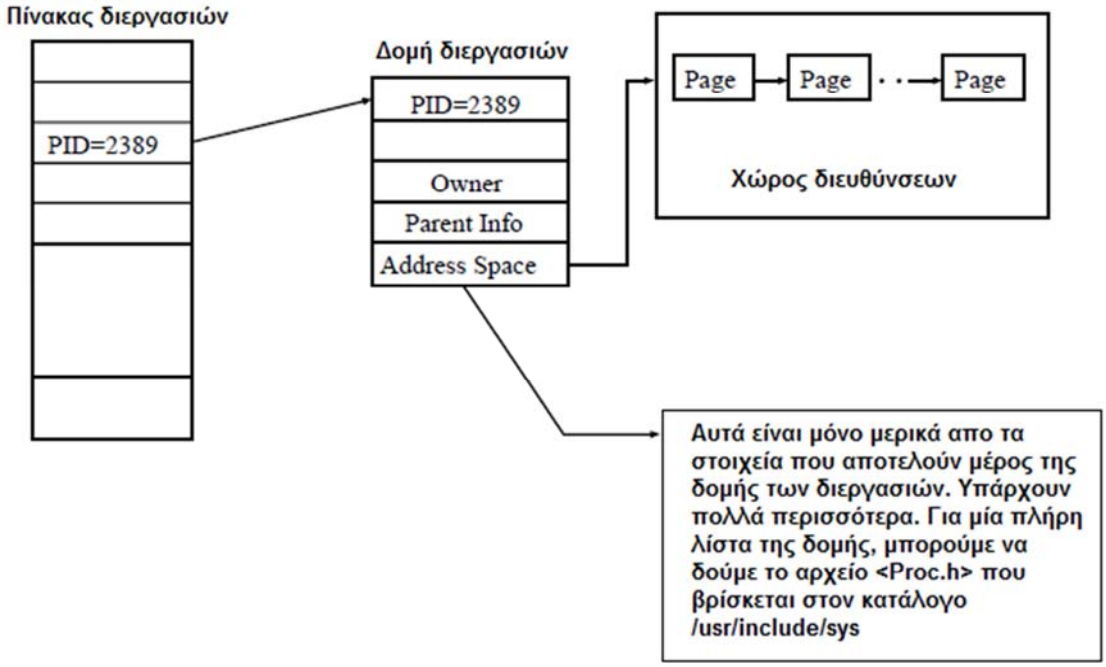
S Η διεργασία βρίσκεται σε κατάσταση ύπνου
O Η διεργασία τρέχει στην CPU
R Η διεργασία είναι έτοιμη να τρέξει
I Η διεργασία βρίσκεται σε αδράνεια; δημιουργείτε τώρα
T Η διεργασία παρακολουθείται ή εντοπίζεται
X Η διεργασία αναμένει πρόσθετη μνήμη
Z Η διεργασία έχει περιέλθει σε κατάσταση Ζόμπι

Σε μια κατάσταση **S**, η διεργασία κοιμάται και περιμένει κάποιο γεγονός. Το γεγονός θα μπορούσε να περιμένει ένα αίτημα ολοκλήρωσης I/O, ή περιμένοντας για ένα κατειλημμένο πόρο, ή να περιμένει κάποια δεδομένα που δεν είναι ακόμα διαθέσιμα. Μια swapped διεργασία θα έχει επίσης μια κατάσταση S (αναμονής). Βάζοντας μια διεργασία σε αναμονή και αργότερα αφυπνίζοντάς την, καταναλώνονται πολλοί πόροι και έτσι μπορεί να οδηγηθούμε σε προβλήματα απόδοσης, ειδικά όταν ένας μεγάλος αριθμός διεργασιών αφυπνιστούν ταυτοχρόνος. Μία κατάσταση **O** σημαίνει ότι η διεργασία είναι ενεργή και λειτουργεί σε μία CPU. Η κατάσταση **R** υποδεικνύει ότι η διεργασία είναι έτοιμη να τρέξει, και μόλις ένας επεξεργαστής καταστεί διαθέσιμος, η διεργασία θα τρέξει (υποθέτοντας ότι δεν προκαταλαμβάνεται από μια υψηλότερης προτεραιότητας διεργασία). Μια κατάσταση **I** υποδηλώνει ότι η διεργασία βρίσκεται σε αδράνεια και ότι δημιουργείτε ή προετοιμάζεται.

Μια κατάσταση **T** υποδεικνύει ότι η διεργασία παρακολουθείται η εντοπίζεται. Αυτό είναι σύνηθες κατά τη διάρκεια λειτουργίας εντοπισμού σφαλμάτων όταν οι γονικές διεργασίες (*parent processes*) εντοπίζουν τις διεργασίες παιδί (*child processes*) για να εντοπίσουν πιθανά προβλήματα. Η **X** αναφέρεται στην κατάσταση κατά την οποία η διεργασία αναμένει πρόσθετη μνήμη. Τέλος, η **Z** αναφέρεται στην κατάσταση ζόμπι. Η κατάσταση ζόμπι έχει δύο βασικές έννοιες: η πρώτη είναι ότι η διεργασία (παιδί) έχει λήξει ή έχει μαρκαριστεί προς τερματισμό και έχει μία γονική διεργασία, και η δεύτερη είναι ότι η διεργασία έχει επισημανθεί προς τερματισμό από τον πυρήνα του UNIX. Κάθε διεργασία έχει μια ολοκληρωμένη δομή που σχετίζεται με αυτή όταν δημιουργείται. Η δομή αυτή έχει μια σύνδεση με το χώρο διευθύνσεων της διεργασίας. Κατά την διάρκεια της κατάστασης ζόμπι, η δομή της διεργασίας και ο χώρος διευθύνσεων αφαιρούνται για να απελευθερωθούν και να επανέλθουν πίσω στο σύστημα. Η μόνη πληροφορία που παραμένει στον πυρήνα για μια διεργασία ζόμπι είναι μια καταχώρηση στον πίνακα διεργασιών. Ο πυρήνας είναι υπεύθυνος για τον καθαρισμό του πίνακα διεργασιών και την αφαίρεση

των καταχωρήσεων/καταστάσεων ζόμπι. Στο σχήμα 2.5 παρουσιάζεται η επισκόπηση της διεργασίας εντός του πυρήνα

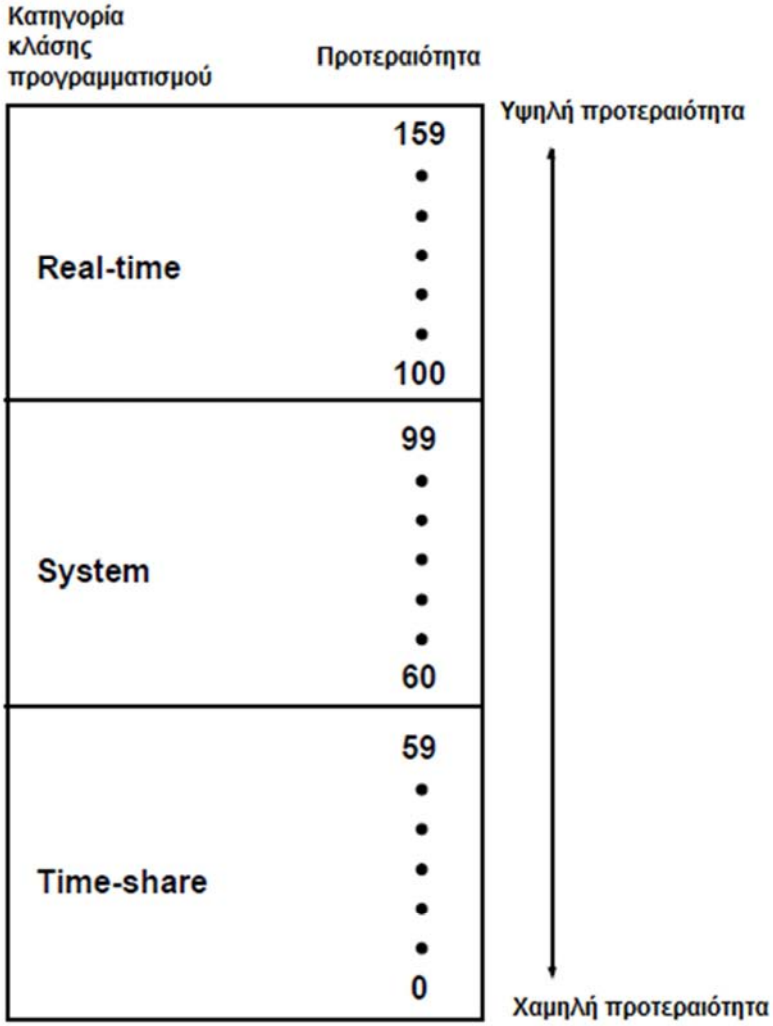
Η δομή διαδικασιών περιλαμβάνει πολλούς διαφορετικούς τομείς, συμπεριλαμβανομένων των πληροφοριών για τις διεργασίες και στατιστικά στοιχεία αυτών. Κάθε φορά που μία διεργασία δημιουργείται, μια ξεχωριστή δομή για τη διεργασία δημιουργείται. Για να δούμε περισσότερα σχετικά με τη δομή της διεργασίας, μπορούμε να διαβάσουμε το αρχείο κεφαλίδας <`Proc.h`> που βρίσκεται στον κατάλογο `/usr/include/sys`. Αυτό το αρχείο παρέχει μια πλήρη λίστα της δομής των διεργασιών.



Σχήμα 2.5 Δομή διεργασιών (Πηγή: pearsonhighered.com)

2.2.4 Ο χρονοπρογραμματιστής των διεργασιών και εργασιών

Ο χρονοπρογραμματιστής των εργασιών και διεργασιών του UNIX παρέχει τρεις διαφορετικούς τύπους κλάσεων εργασίας (***Inside the unix kernel***, 2000) όπως παρατίθενται στον Πίνακα 2.3. Κάθε κλάση που αναφέρεται στον πίνακα 2.3 έχει ένα διακριτό σύνολο γενικής προτεραιότητας που ελέγχει τη σειρά προγραμματισμού των εργασιών (διεργασίες) εντός της συγκεκριμένης κατηγορίας. Οι κλάσεις των εργασιών αναφέρονται στον πίνακα 2.3 κατά αύξουσα σειρά προτεραιότητας. Είναι κρίσιμο να κατανοήσουμε τον σκοπό καθώς και τις ιδιότητες της κάθε κλάσης . Η κατανόηση των διαφόρων κατηγοριών κλάσεων θα μας δώσει τη δυνατότητα για την περαιτέρω κατανόηση της πολιτικής που ακολουθεί ο πυρήνας σε σχέση με τον προγραμματισμό των εργασιών. Πρέπει να τονίσουμε ότι πρέπει να λαμβάνεται ιδιαίτερη προσοχή πριν από την αλλαγή της κατηγορίας κλάσης ή / και της προτεραιότητας κάποιας συγκεκριμένης διεργασίας. Με αυτόν τον τρόπο χωρίς να έχουμε πλήρη κατανόηση των πιθανών επιπτώσεων θα μπορούσαμε να προκαλέσουμε αστάθεια στο σύστημα, και ενδεχομένως να αναγκάζαμε το σύστημα σε απορύθμιση. Το σχήμα 2.6 συνοψίζει τον προγραμματισμό εργασιών του UNIX.



Σχήμα 2.6 Προγραμματισμός εργασιών και διεργασιών στο Unix (Πηγή: pearsonhighered.com)

Κατηγορία κλάσης	Εύρος προτεραιότητας
Time-share (προεπιλεγμένη κλάση)	0 έως 59 (ανώτερη τιμή)
System	60 έως 99 δεσμευμένο για τις διεργασίες του δαίμονα
Real-time	100 έως 159 για τις διαδικασίες υψηλής προτεραιότητας

Πίνακας 2.3 Κατηγορίες κλάσεων που παρέχονται από τον προγραμματιστή διεργασιών του Unix (Πηγή: pearsonhighered.com)

2.2.4.1 Η κλάση Time-share

Η κλάση Time-share είναι η προεπιλεγμένη κλάση και έχει προτεραιότητα από 0 έως 59, με το 59 να αποτελεί την υψηλότερη προτεραιότητα. Σε αυτή την κατηγορία κλάσης, σε κάθε διεργασία έχει εκχωρηθεί ένας διαμοιραζόμενος χρόνος. Ο χρόνος αυτός καθορίζει τον αριθμό των χτύπων του ρολογιού της CPU που μία συγκεκριμένη διεργασία μπορεί να καταλάβει (με την έννοια του καταλαμβάνω) στην CPU. Μόλις ο χρόνος, που έχει εκχωρηθεί στην διεργασία τελειώσει, η προτεραιότητα της διεργασίας συνήθως μειώνεται και η διεργασία τοποθετείται στην ουρά αναμονής. Οι άλλες διεργασίες αναμένουν την προτεραιότητα τους να αυξηθεί από την CPU και από την μείωση προτεραιότητας που έχει επέλθει σε κάποιες διεργασίες, εκτός και αν ορίζεται διαφορετικά μέσω των εντολών dispadmin[1] ή priocntl, κάθε φορά που μια νέα διεργασία δημιουργείτε θα πρέπει να ανατίθεται στην κλάση Time-share με ένα προεπιλεγμένο επίπεδο προτεραιότητας. Μπορούμε να χρησιμοποιήσουμε την εντολή `ps -c` για να δούμε τη λίστα των διαφόρων κλάσεων και την προτεραιότητα των διεργασιών. Για παράδειγμα,

Myhost%> ps -c

PID	CLS	PRI	COMD
2403	TS	59	csh
15231	TS	48	ps

Σε αυτό το παράδειγμα, η έξοδος από την εντολή `ps -c` δείχνει το αναγνωριστικό της διεργασίας (PID) του τρέχοντος κελύφους όπως επίσης και την ίδια την εντολή `ps`. Και οι δύο διεργασίες είναι στην κλάση Time-share (TS), και έχουν μια προτεραιότητα 59 ή λιγότερο (PRI).

2.2.4.2 Η κλάση System

Η κλάση System είναι δεσμευμένη για τις διεργασίες δαίμονα[2] (daemon) όπως η pageout ή ο δαίμονας του συστήματος αρχείων. Το εύρος προτεραιότητας για την κλάση System είναι μεταξύ του 60 και του 99. Η κλάση System έχει υψηλότερη προτεραιότητα από τη Time-share. Αυτό γίνεται προκειμένου να εξασφαλίζεται ότι το σύστημα μπορεί να παρέχει υπηρεσίες στις διεργασίες κατηγορίας Time-share, όταν ζητούνται οι υπηρεσίες του συστήματος όπως είναι η μνήμη, τα αρχεία κλπ. Αν και είναι δυνατόν να μεταβάλλουμε την κλάση Time-share μιας διαδικασίας σε κλάση Real-time, δεν συνιστάται, διότι η διεργασία θα τρέξει σε υψηλότερη προτεραιότητα από τους δαίμονες του συστήματος. Πρέπει να είμαστε ιδιαίτερα προσεκτικοί πριν αλλάξουμε την τάξη ή την προτεραιότητα των διεργασιών.

[1]Η εντολή dispadmin είναι μία βοηθητική εντολή/εφαρμογή του Solaris που χρησιμοποιείται για τη διαχείριση του χρονοδιαγράμματος των διαδικασιών, ενώ το σύστημα τρέχει. Η εντολή αυτή χρησιμοποιείτε για να αλλάξουμε την κατηγορία κλάσης

[2]Σε ένα περιβάλλον Unix, η γονική διεργασία ενός δαίμονα είναι συχνά, αλλά όχι πάντα, η διαδικασία init. Ένας δαίμονας δημιουργείτε συνήθως είτε από μια διαδικασία forking (child process) και αμέσως μετά εξάγεται, προκαλώντας έτσι την init να υιοθετήσει τη διαδικασία παιδιού, ή η διαδικασία init εκκινεί απευθείας τον δαίμονα.

2.2.4.3 Η κλάση Real-time

Η κλάση Real-time είναι η υψηλότερη κλάση προτεραιότητας και δεν διαμοιράζει καθόλου τον χρόνο στις διάφορες διεργασίες. Ο πραγματικός χρόνος προτεραιότητας της διεργασίας κυμαίνονται μεταξύ του 100 και του 159. Η κατηγορία αυτή εξασφαλίζει ότι οι κρίσιμες διεργασίες αποκτούν πάντα πόρους από τη CPU μόλις η διεργασία έχει προγραμματιστεί να τρέξει, ακόμα και αν οι υπόλοιπες διεργασίες στις άλλες κλάσεις είναι ήδη σε εξέλιξη ή έχουν προγραμματιστεί να τρέξουν. Και πάλι θα πρέπει να είμαστε προσεκτικοί κατά την αλλαγή των προτεραιοτήτων ή των κατηγοριών των κλάσεων. Μπορεί να υπάρχουν κάποια λειτουργικά συστήματα που να αντιμετωπίζουν κάποια θέματα, όταν η τάξη ή η προτεραιότητα μιας διεργασίας μεταβληθεί. Επειδή δεν υπάρχει διαμοιραζόμενος χρόνος στην συγκεκριμένη κατηγορία κλάσης, ο αριθμός των context switches μειώνεται σημαντικά, το οποίο μπορεί να αυξήσει την απόδοση του συστήματος.

2.2.5 Σήματα και διακοπές

Θεμελιώδους σημασίας για οποιοδήποτε λειτουργικό σύστημα είναι οι έννοιες των σημάτων και των διακοπών η οποία αναλύθηκε και γενικά στην παράγραφο 1.5.1 του πρώτου κεφαλαίου (***Inside the unix kernel***, 2000). Τα σήματα είναι συμβάντα/γεγονότα του λογισμικού και οι διακοπές είναι συμβάντα/γεγονότα του υλικού. Τα σήματα μπορεί να είναι ασύγχρονα ή σύγχρονα. Ασύγχρονα σήματα μπορούν να εμφανιστούν σε οποιοδήποτε σημείο. Ασύγχρονα σήματα είναι τα γεγονότα όπως το σήμα `QUIT (CTRL-\)`, το σήμα διακοπής από το λογισμικό `(CTRL-C)`, ή το κατειλημμένο σήμα. Τα σύγχρονα σήματα προκαλούνται από μια μη αποδεκτή λειτουργία, όπως μια κινητή υποδιαστολή (εξαίρεση αυτής), ένα σφάλμα διαίρεσης με το μηδέν, μία παραβίαση κατάτμησης, ή ένα λάθος διαύλου. Τα σήματα χρησιμοποιούνται συνήθως εντός των εφαρμογών ως μέθοδος επικοινωνίας μεταξύ των διεργασιών.

Οι εφαρμογές επίσης χρησιμοποιούν τα σήματα για να παγιδεύσουν ορισμένα γεγονότα, όπως για παράδειγμα ένα γεγονός που προέρχεται από τον χρήστη ο οποίος τερματίζει μία συγκεκριμένη διαδικασία. Η εφαρμογή θα μπορούσε να παγιδεύσει το σήμα τερματισμού και στη συνέχεια να τερματίσει με περισσότερη ευκολία, απαλλαγμένη από πρόσθετες διεργασίες. Μπορούμε να χρησιμοποιήσουμε τις κλήσεις συστήματος `signal-()`, `sigset-()`, ή `sigaction-()` για να δημιουργήσουμε ένα χειριστή σήματος. Μπορούμε επίσης να επιλέξουμε να γίνεται μια συγκεκριμένη ενέργεια κατά τη λήψη ενός σήματος ή να επιλέξουμε να αγνοήσουμε το σήμα και να επιτρέψουμε στην αίτησή μας να συνεχίσει. Ορισμένα σήματα, όπως μοιραία συμβάντα για το υλικό, δεν μπορούν να αγνοηθούν και συνήθως καταλήγουν να απορρίπτονται και την εφαρμογή να τερματίζετε.

Οι διακοπές είναι γεγονότα του πυρήνα που χρησιμοποιούνται για να ενημερώσουν τον πυρήνα, όταν ένα συμβάν έχει συμβεί. Μια εφαρμογή χρήστη μπορεί να παράγει μια σειρά από κλήσεις I/O και ο πυρήνας να εκχωρήσει αυτές τις κλήσεις προς την ουρά I/O. Οι συσκευές I/O που εμπλέκονται, θα επεξεργαστούν στη συνέχεια τις αιτήσεις I/O που ζητήθηκαν και έπειτα θα στείλουν μια διακοπή στον πυρήνα, κοινοποιώντας του επίσης ότι οι αιτήσεις I/O έχουν ολοκληρωθεί. Ο πυρήνας

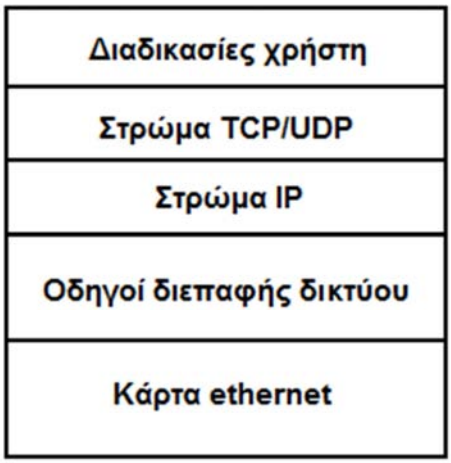
μπορεί στη συνέχεια να στείλει ένα μήνυμα προς τη διαδικασία χρήστη ότι οι αιτήσεις I/O έχουν ολοκληρωθεί. Η διαδικασία χρήστη μπορεί έπειτα να ελέγξει τον κωδικό σφάλματος της κλήσεις συστήματος I/O έτσι ώστε να καθορίσει εάν οι αιτήσεις I/O έχουν ολοκληρωθεί με επιτυχία. Ο πυρήνας χρησιμοποιεί επίσης διακοπές για να προκαταλάβει τις διεργασίες ή να βγάλει ένα επεξεργαστή off-line.

2.2.6 Υπηρεσίες δικτύου και επικοινωνίες

Ο πυρήνας παρέχει διάφορες υπηρεσίες δικτύου (***Inside the unix kernel***, 2000), συμπεριλαμβανομένων της απομακρυσμένης σύνδεσης, της μεταφοράς αρχείων και του NFS (Network File System). Ο πυρήνας παρέχει το σύστημα δαίμονα inetd[1], για την παροχή υπηρεσιών δικτύου, όπως rlogin[2], rsh[3], ftp, και πολλές άλλες υπηρεσίες που βασίζονται στα TCP/IP. Η προεπιλεγμένη οικογένεια πρωτοκόλλων για τον πυρήνα του UNIX είναι το TCP/IP.

Όταν ξεκινάει το σύστημα σε κατάσταση λειτουργίας multi-user, ο δαίμονας inetd ξεκινάει και αυτός. Ο inetd είναι ο δαίμονας υπηρεσιών διαδικτύου ο οποίος είναι υπεύθυνος για τις υπηρεσίες telnet, rlogin, rsh, και άλλους τύπους υπηρεσιών διαδικτύου. Έπειτα από ένα login ή telnet, ο inetd με την βοήθεια της διεργασίας `fork-()` εκτελεί τον κατάλληλο δαίμονα, όπως είναι ο in.telnetd για μια συνεδρία telnet. Το αρχείο `/etc/inet/inetd.conf` είναι αυτό που χρησιμοποιείται για την ρύθμιση των υπηρεσιών του inetd στο σύστημα του solaris. Μπορούμε να χρησιμοποιήσουμε αυτό το αρχείο για να περιορίσουμε ορισμένες υπηρεσίες του Internet, όπως το απομακρυσμένο κέλυφος (rsh) ή το finger. Μετά από ένα αίτημα υπηρεσίας Internet, όπως το FTP, ο inetd με την βοήθεια της fork καλεί τον κατάλληλο δαίμονα υπηρεσιών, όπως είναι ο in.ftpd για την περίπτωση μιας αίτησης FTP. Χρησιμοποιώντας την εντολή ps, μπορούμε να το επιβεβαιώσουμε, καθώς θα πρέπει να υπάρχει μια διαδικασία για την ftp (In.ftpd), και το αναγνωριστικό της διεργασίας του γονέα θα πρέπει να είναι το ID της διεργασίας του inetd δαίμονα. Οι περισσότεροι διαχειριστές συστημάτων απενεργοποιούν ορισμένες υπηρεσίες Internet, όπως το απομακρυσμένο κέλυφος και το finger, προκειμένου να αυξήσουν την ασφάλεια του συστήματός τους.

[1]Η inetd (internet service daemon) είναι ένας δαίμονας υπερ-server σε πολλά συστήματα Unix που παρέχει υπηρεσίες Internet.
[2]Το rlogin είναι ένα βοηθητικό πρόγραμμα για Unix-like λειτουργικά συστήματα που επιτρέπει στους χρήστες να συνδεθούν σε έναν άλλο υπολογιστή μέσω δικτύου, η επικοινωνία γίνεται μέσω της θύρας TCP 513.
[3]Το απομακρυσμένο κέλυφος (rsh) είναι ένα πρόγραμμα της γραμμής εντολών που μπορεί να εκτελέσει εντολές για έναν απομακρυσμένο υπολογιστή (μέσω δικτύου), ως άλλος χρήστης. Το απομακρυσμένο σύστημα στο οποίο το rsh συνδέεται καλεί τον δαίμονα rsh (rshd). Ο δαίμονας συνήθως χρησιμοποιεί το γνωστό πρωτόκολλο (TCP) και την θύρα 514.



Σχήμα 2.7 Στρώματα δικτύου

2.2.7 Συγχρονισμός

Ο πυρήνας του UNIX παρέχει αρκετές διαφορετικές μεθόδους συγχρονισμού ή ταυτοχρονισμού (concurrency) (***Inside the unix kernel***, 2000). Ο συγχρονισμός είναι η διαδικασία της δέσμευσης ενός πόρου για να εξασφαλιστεί η συνοχή, έτσι ώστε δύο ή περισσότερες διεργασίες να μην ακυρώνουν η μία τις τροποποιήσεις της άλλης. Παρά το γεγονός ότι πολλές διεργασίες μπορούν να εκτελούνται ταυτόχρονα, οι δεσμεύσεις πόρων είναι απαραίτητες όταν μια διεργασία θέλει να προστατεύσει έναν πόρο αρχείου, δεδομένα, ή / και το τμήμα της μνήμης. Για παράδειγμα, μπορούμε να σκεφτούμε μία βασική τραπεζική συναλλαγή: Ένα άτομο είναι στην διαδικασία ανάληψης χρημάτων από το λογαριασμό του, ενώ ένας άλλος συγγενής καταθέτη κάποιο χρηματικό ποσό στον ίδιο λογαριασμό ταυτόχρονα. Χωρίς συγχρονισμό (δέσμευση/κλείδωμα), στο τέλος της συναλλαγής το υπόλοιπο του λογαριασμού μπορεί να μην εμφανίζεται σωστά.

Ο πυρήνας Solaris παρέχει τις ακόλουθες μεθόδους συγχρονισμού:

1. κλειδαριές (locks) mutex
2. μεταβλητές κατάστασης (condition variables)
3. σηματοφορείς (semaphores)
4. κλειδαριές πολλαπλών αναγνωστών (multi-reader) και μονών εγγραφέων (single-writer locks)

Οι Mutex είναι κλειδαριές αμοιβαίου αποκλεισμού και μπορούν να χρησιμοποιηθούν στο επίπεδο της επεξεργασίας των διεργασιών. Οι Mutex κλειδαριές είναι απλές και αποτελεσματικές κλειδαριές χαμηλού επιπέδου. Οι μεταβλητές κατάστασης χρησιμοποιούνται σε συνδυασμό με τις mutex κλειδαριές για να επαληθεύουν ή να περιμένουν μια συγκεκριμένη κατάσταση. Οι σηματοφόροι είναι μια τυποποιημένη μέθοδος συγχρονισμού και αποτελούν μέρος του συστήματος V έκδοσης 4. Ένας σηματοφόρος είναι ένας μη αρνητικός ακέραιος αριθμός καταμέτρησης των διαθέσιμων πόρων. Ο ακέραιος αριθμός είναι είτε αυξανόμενος ή

μειούμενος, καθώς οι πόροι ελευθερώνονται ή λαμβάνονται, αντίστοιχα. Ένας σηματοφόρος που ξεκινάει την καταμέτρηση εκ του μηδενός δείχνει ότι δεν υπάρχουν άλλοι πόροι διαθέσιμοι. Οι πολλαπλοί αναγνώστες και οι εγγραφείς είναι πιο περίπλοκοι από ό, τι οι άλλοι τρεις τύποι συγχρονισμού καθώς παρέχουν κλειδαριές πολλαπλών αναγνώσεων καθώς και μία κλειδαριά εγγραφέα.

2.2.8 Οι δαίμονες του συστήματος

Η έννοια των δαιμόνων είναι θεμελιώδης σημασίας για το UNIX (**documentation of freebsd** 2016**, *Inside the unix kernel*,** 2000). Ένας δαίμονας είναι απλά, όπως ο όρος προτείνει, μία εργασία που ενεργεί για λογαριασμό ενός άλλου έργου ή οντότητας. Με άλλα λόγια, ένας δαίμονας δεν είναι τίποτα περισσότερο από μια υποβοηθούσα παρασκηνιακή δουλειά ή διαδικασία που είναι υπεύθυνη για μια συγκεκριμένη εργασία ή ένα σύνολο εργασιών. Ο πυρήνας του UNIX αποτελείται από πολυάριθμους δαίμονες συστήματος που είναι υπεύθυνοι για τη διαχείριση της μνήμης, τη διαχείριση των αρχείων του συστήματος, τις εργασίας εκτύπωσης, τις συνδέσεις δικτύου και πολλές άλλες υπηρεσίες. Για παράδειγμα, ας δούμε την έξοδο που φαίνεται στον Πίνακα 2.4 από την εντολή `ps`.

Η λίστα δεδομένων του Πίνακα 2.4 από την εντολή `ps` εμφανίζει μερικούς από τις δαίμονες οι οποίοι τρέχουν. Ο δαίμονας pageout (στον πίνακα 2.4 με ID διεργασίας = 2) είναι υπεύθυνος για την τροποποίηση των εγγραφών στις σελίδες της μνήμης του συστήματος αρχείων, για την μεταφορά των σελίδων μεταξύ των ουρών για καλύτερη ισορροπία της μνήμης και των συγχρονισμό τους. Ο κύριος στόχος του δαίμονα page είναι να διασφαλίσει ότι στο σύστημα θα υπάρχει επαρκείς μνήμη για τις εργασίες και διαδικασίες που χρειάζεται να γίνουν.

Ένας άλλος δαίμονας του συστήματος είναι ο fsflush. Αυτός ο δαίμονας είναι υπεύθυνος για την εγγραφή των δεδομένων από το buffer στο δίσκο. Αυτή η εντολή χρησιμοποιείται για τις λειτουργίες του διακομιστή αρχείων. Ο πυρήνας του UNIX χρησιμοποιεί ένα σύστημα αρχείων UNIX cache για τη διαχείριση του συστήματος αρχείων. Όλες οι εγγραφές που γίνονται και βασίζεται στο σύστημα αρχείων γίνονται πρώτα στην μνήμη cache (γι αυτό προσέχουμε το υλικό του υπολογιστή μας να διαθέτει όση περισσότερη μνήμη cache). Οι δαίμονες είναι ζωτικής σημασίας για το λειτουργικό σύστημα και τις υπηρεσίες που αιτούνται τόσο από τον πυρήνα όσο και από τον χρήστη. Ο στόχος αυτών των δαιμόνων είναι να ικανοποιήσουν αυτά τα αιτήματα, ενώ θα πρέπει να διατηρούν παράλληλα και την ακεραιότητα του συστήματος.

UID	PID	PPID	TIME CMD
root	0	0	0:01 sched
root	1	0	0:01 /etc/init -
root	2	0	0:00 pageout
root	3	0	3:27 fsflush
root	131	1	0:01 /usr/sbin/inetd -s
root	289	1	0:00 /usr/lib/saf/sac -t 300
root	112	1	0:02 /usr/sbin/rpcbind
root	186	1	0:00 /usr/lib/lpsched
root	104	1	0:12 /usr/sbin/in.routed -q
root	114	1	0:01 /usr/sbin/keyserv
root	122	1	0:00 /usr/sbin/kerbd
root	120	1	0:00 /usr/sbin/nis_cachemgr
root	134	1	0:00 /usr/lib/nfs/statd
root	136	1	0:00 /usr/lib/nfs/lockd
root	155	1	0:00 /usr/lib/autofs/automountd
root	195	1	0:00 /usr/lib/sendmail -bd -q1h
root	159	1	0:00 /usr/sbin/syslogd
root	169	1	0:13 /usr/sbin/nscd
root	194	186	0:00 IpNet

Πίνακας 2.4 Αποτελέσματα της εντολής ps (Πηγή: pearsonhighered.com)

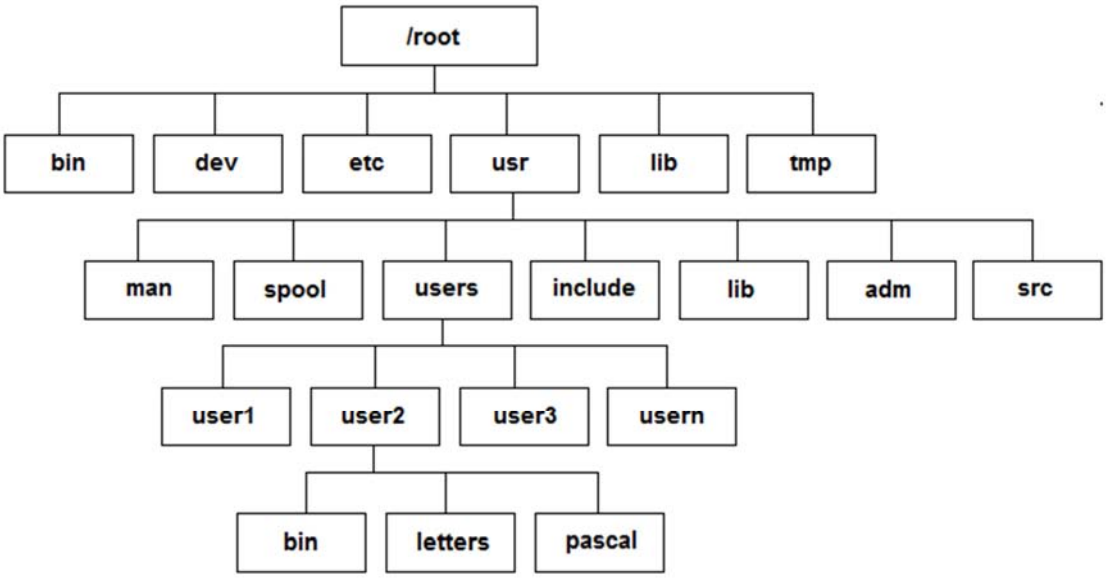
2.2.9 Τα κελύφη (shells)

Το λειτουργικό σύστημα UNIX παρέχει επίσης διάφορους τύπους κελυφών που δρουν σαν διερμηνείς εντολών (***Inside the unix kernel***, 2000). Το UNIX παρέχει τρία καθιερωμένα είδη κελυφών: Το Bourne (sh), το Korn (ksh), και το C-shell (csh). Το Bourne Shell και το C-shell είναι τα πιο κοινά. Μπορούμε να χρησιμοποιήσουμε το κέλυφος της επιλογής μας και να το προσαρμόσουμε ανάλογα με τη χρήση των (.) αρχείων όπως τα .login, .cshrc, και .logout για το C-shell. Το .cshrc είναι το αρχείο των πόρων του Cshell κάθε φορά που ένα C-shell δημιουργείται. Μπορούμε να χρησιμοποιήσουμε τα κελύφη έτσι ώστε να επικαλούνται συγκεκριμένα σενάρια ή προγράμματα εκκίνησης. Το προεπιλεγμένο κέλυφος του συστήματος μπορεί να ρυθμιστεί από το διαχειριστή του UNIX στο αρχείο `/etc/passwd`. Ορισμένες εκδόσεις του UNIX παρέχουν ένα περιορισμένο περιβάλλον κελύφους (`/usr/lib/rsh`). Αυτό δεν πρέπει να συγχέεται με το πρότυπο απομακρυσμένο κέλυφος (rsh), το οποίο είναι για απομακρυσμένη σύνδεση. Το περιορισμένο κέλυφος είναι ένα ειδικό κέλυφος που περιορίζει το χρήστη από το να αλλάξει κάποιες από τις μεταβλητές περιβάλλοντος και από την αλλαγή των τοποθεσιών στον κατάλογο. Κάτω από το περιορισμένο κέλυφος, ο χρήστης μπορεί να εκτελέσει μόνο εντολές που βρίσκονται στη μεταβλητή $PATH[1]. Αυτό το κέλυφος είναι συχνά χρήσιμο για ασφαλείς λογαριασμούς χρηστών που εκτελούν μόνο ορισμένες εφαρμογές. Ο διαχειριστής του συστήματος μπορεί να χρησιμοποιήσει ένα περιορισμένο περιβάλλον εντολών για το κέλυφος με σκοπό την αύξηση τις ασφάλειας του συστήματος, επιτρέποντας στο χρήστη να τρέξει μόνο ορισμένες εντολές και εφαρμογές.

[1]Το PATH είναι μια μεταβλητή για περιβάλλοντα Unix-like, DOS, OS/2, και τα MS Windows, προσδιορίζοντας ένα σύνολο καταλόγων, όπου βρίσκονται εκτελέσιμα προγράμματα.

2.2.10 Το σύστημα αρχείων

Το σύστημα αρχείων του Unix έχει τη μορφή ενός αντεστραμμένου δέντρου (**Αραποστάθης Μ.**, 2011). Στη ρίζα του δέντρου υπάρχει ένας κατάλογος που λέγεται root και συμβολίζεται με /. Κάθε κόμβος του δέντρου είναι κατάλογος, και μέσα σε κάθε κατάλογο μπορούν να περιέχονται κανονικά και ειδικά αρχεία, και φυσικά άλλοι κατάλογοι. Με την εγκατάσταση του Unix σε ένα υπολογιστικό σύστημα δημιουργείται ο κατάλογος /, καθώς και κάποιοι άλλοι, οι κυριότεροι των οποίων φαίνονται στο σχήμα 2.8 και επεξηγούνται στη συνέχεια στον πίνακα 2.5.



Σχήμα 2.8 Αποτύπωση του συστήματος αρχείων

Τα περιεχόμενα αυτών των καταλόγων είναι τα ακόλουθα:

/bin	Συχνά χρησιμοποιούμενα εκτελέσιμα αρχεία του συστήματος
/dev	Ειδικά αρχεία για συσκευές εισόδου-εξόδου
/etc	Διάφορα αρχεία για τη διαχείριση του συστήματος
/lib	Συχνά χρησιμοποιούμενες βιβλιοθήκες
/tmp	Μερικά προγράμματα δημιουργούν εδώ προσωρινά αρχεία
/usr/adm	Αρχεία λογιστικής του συστήματος (accounting)
/user/include	Αρχεία ενσωμάτωσης συστήματος
/usr/man	Αρχεία άμεσης βοήθειας
/usr/spool	Κατάλογοι φύλαξης ετεροχρονισμένης εισόδου – εξόδου
/usr/src	Πρωτογενής κώδικας του συστήματος
/usr/user2	Ατομικός κατάλογος του χρήστη με όνομα user2
/usr/user2/pascal	Κατάλογος που έχει δημιουργήσει ο user2

Πίνακας 2.5 Εξήγηση και περιεχόμενα καταλόγων του συστήματος αρχείων

Σε κάθε χρήστη που συνδέεται στο σύστημα παραχωρείται ένας κατάλογος (συνήθως κάτω από τον usr/users) που έχει σαν όνομα τον κωδικό εισόδου του χρήστη και λέγεται ατομικός κατάλογος (home directory) του χρήστη. Ο χρήστης μπορεί να κάνει ό, τι θέλει στον ατομικό του κατάλογο (δηλ. να δημιουργήσει πολλαπλά επίπεδα υποκαταλόγων, να φτιάξει και να διαγράψει αρχεία κ.λ.π.). Αν και συνήθως μπορεί να μετακινηθεί σε καταλόγους πάνω από τον ατομικό του κατάλογο, τα δικαιώματα του εκεί περιορίζονται.

Το Unix χρησιμοποιεί ορισμένα σύμβολα για να παραστήσει ειδικούς καταλόγους. Αυτά είναι τα εξής:

/ Συμβολίζει το βασικό κατάλογο (ρίζα) και διαχωρίζει τους κατάλογους μίας διαδρομής
. Ο τρέχων κατάλογος (εργασίας)
.. Ο πατρικός ενός καταλόγου
~ Ο ατομικός κατάλογος (εκκίνησης)

Σε ένα αρχείο ή σε ένα κατάλογο μπορούμε να αναφερθούμε με δύο τρόπους. Με το πλήρες όνομα διαδρομής (full path name) ή με το σχετικό όνομα διαδρομής (relative path name). Όταν χρησιμοποιούμε το πλήρες όνομα διαδρομής προσδιορίζουμε τη θέση του καταλόγου ή αρχείου από το βασικό κατάλογο /. Π.χ. με το /usr/users/user2/pascal/calc.p αναφερόμαστε στο αρχείο c.p ξεκινώντας από τον βασικό κατάλογο.

Η προσέγγιση αυτή δεν είναι πάντα η βολικότερη. Π.χ. αν ο τρέχων κατάλογος είναι ο /users/user1 τότε μπορούμε να αναφερθούμε στο αρχείο calc.p χρησιμοποιώντας την παράσταση ..user2/pascal/calc.p διατρέχοντας δηλαδή τους καταλόγους όχι από τον βασικό αλλά από τον τρέχοντα κατάλογο. Όταν κάνουμε αυτό λέμε ότι χρησιμοποιούμε σχετικό όνομα διαδρομής.

2.3 Εκδόσεις του Unix και σύγκριση με το Linux

Ένας τεράστιος αριθμός εκδόσεων οι οποίες στηρίζονται στο Unix ή έχουν παρόμοια δομή και κατευθύνσεις, συνυπάρχουν και καθημερινά γίνονται όλο και περισσότερες. Αξίζει να σημειωθεί πώς το GNU/Linux δεν μπορεί να χαρακτηριστεί ακριβώς ως Unix σύστημα αλλά ως ένα Unix-like δηλαδή ένα λειτουργικό σύστημα που είναι παρόμοιο με το Unix αλλά διαφέρει σε αρκετά στοιχεία, εφαρμογές, άδειες κλπ. Στον πίνακα 2.6 θα προσπαθήσουμε να αποτυπώσουμε τις βασικές διαφορές των δύο αυτών συστημάτων καθώς υπάρχει μία μεγάλη σύγχυση σε σχέση με το τι είναι Unix και το τι είναι τελικά το GNU/Linux. Κάποια βασικά στοιχεία του Unix που θα πρέπει να έχουμε στα υπόψη μας είναι τα παρακάτω:

- ✓ Το UNIX είναι ένα κατοχυρωμένα όνομα που μόνο μεγάλες εταιρείες επιτρέπεται να χρησιμοποιούν τα πνευματικά του δικαιώματα και το όνομά του
- ✓ Έτσι το σύστημα IBM AIX, το Sun Solaris, και το HP-UX είναι όλα λειτουργικά συστήματα UNIX.

- ✓ Το μεγαλύτερο και πιο δημοφιλές λειτουργικό σύστημα Unix για επιτραπέζιους υπολογιστές είναι το Apple OS X
- ✓ Η εταιρία Open Group[1] κατέχει το εμπορικό σήμα UNIX, και διαχειρίζεται το πρόγραμμα αδειοδότησης του εμπορικού σήματος.
- ✓ Τα περισσότερα συστήματα UNIX έχουν εμπορικό χαρακτήρα.
- ✓ Αυτό βέβαια δεν είναι απόλυτο καθώς η Oracle με το open Solaris που είναι και το πρώτο ελεύθερο σύστημα Unix κατέστησε δυνατή την εγκατάσταση ενός Unix συστήματος χωρίς την προϋπόθεση δικαιωμάτων και εμπορικού χαρακτήρα μέχρι την στιγμή που σταμάτησε την υποστήριξή του (βέβαια το εγχείρημα συνεχίστηκε από τον έργο openindiana το οποίο αναφέρεται στη συνέχεια).

Πίνακας 2.6 Βασικές διαφορές μεταξύ Unix και Linux (πηγή: www.diffen.com)

	Unix	Linux
Κόστος	Διάφορες εκδόσεις του Unix έχουν διαφορετικά κόστη και δομή, ανάλογα με τον κατασκευαστή	Το Linux διανέμεται δωρεάν, μπορούμε να κατεβάσουμε το αρχείο προς εγκατάσταση από διαδικτυακούς τόπους κλπ. Υπάρχουν και επί πληρωμή εκδόσεις οι οποίες όμως είναι φθηνότερες σε σχέση με τα Windows
Ανάπτυξη και διανομή	Τα συστήματα του Unix χωρίζονται σε διάφορες κατηγορίες, κυρίως αναπτύχθηκε από την AT&T καθώς και διάφορες εμπορικές επιχειρήσεις και πωλητές όπως και μη-κερδοσκοπικούς οργανισμούς.	Το Linux έχει αναπτυχθεί από την Open Source development, μέσω της ανταλλαγής και της συνεργασίας του κώδικα και διατίθεται μέσω φόρουμς κλπ. Διανέμεται από διάφορους προμηθευτές.
Κατασκευαστές	Οι τρεις μεγαλύτερες διανομές είναι το Solaris (Oracle), το AIX (IBM) και το HP-UX Hewlett Packard. Επίσης η Apple κατασκευάζει το OS X, ένα λειτουργικό σύστημα βασισμένο στο Unix	Ο kernel των Linux έχει αναπτυχθεί από την κοινότητα που ασχολείτε με την συντήρηση του. Ο Linus Torvalds επιβλέπει τις εργασίες.
Χρήστης	Το λειτουργικό σύστημα Unix αναπτύχθηκε κυρίως για υπερυπολιστές, servers και σταθμούς εργασίας εκτός του OS X, το οποίο έχει σχεδιαστεί για όλους. Το περιβάλλον του Unix και το πρότυπο πρόγραμμα client-server αποτέλεσαν βασικά στοιχεία στην ανάπτυξη του διαδικτύου	Όλοι μπορούν να χρησιμοποιήσουν το Linux, από οικιακούς χρήστες και προγραμματιστές μέχρι και εταιρίες και μεγάλες επιχειρήσεις

Χρήση	Το λειτουργικό σύστημα UNIX χρησιμοποιείται στους internet servers, σε σταθμούς εργασίας και σε υπολογιστές.	Το Linux μπορεί να εγκατασταθεί σε ένα ευρύ φάσμα υλικού υπολογιστών, που μπορεί να είναι από κινητά τηλέφωνα, υπολογιστές και tablet, κονσόλες βιντεοπαιχνιδιών μέχρι mainframes και υπερυπολογιστές.
Υποστήριξη συστήματος αρχείων	jfs, gpfs, hfs, hfs+, ufs, xfs, zfs	Ext2, Ext3, Ext4, Jfs, ReiserFS, Xfs, Btrfs, FAT, FAT32, NTFS
Διεπαφή εντολών (CLI ή Shell)	Αρχικά υπήρχε μόνο ο Bourne Shell. Τώρα είναι συμβατό με πολλά άλλα κελύφη, συμπεριλαμβανομένων των BASH, Korn & C.	Το BASH (Bourne Again Shell) είναι το προεπιλεγμένο κέλυφος του Linux. Μπορεί να υποστηρίξει πολλαπλές διερμηνείς εντολών. Μπορούμε να χρησιμοποιήσουμε όποιο κέλυφος θέλουμε απο τα πάμπολλα που υπάρχουν
Τί είναι	Το Unix είναι ένα λειτουργικό σύστημα που είναι πολύ δημοφιλές στα πανεπιστήμια, στις εταιρείες, στις μεγάλες επιχειρήσεις κ.λπ.	Το Linux είναι ένα παράδειγμα ανάπτυξης λογισμικού ανοικτού κώδικα και δωρεάν λειτουργικού συστήματος (OS).
Γραφικό περιβάλλον (GUI)	Αρχικά το Unix ήταν ένα λειτουργικό σύστημα που βασίζονταν στις εντολές, αλλά αργότερα ένα GUI δημιουργήθηκε που ονομάζονταν Common Desktop Environment. Οι περισσότερες διανομές Unix σήμερα έρχονται με το Gnome.	Το Linux παρέχει συνήθως δύο επιλογές γραφικού περιβάλλοντος, το KDE και το Gnome. Υπάρχουν βέβαια πολλές άλλες εναλλακτικές επιλογές, όπως το LXDE, Xfce, Cinnamon, Unity, Mate, twm, κλπ.
Τιμή	Κάποιες διανομές είναι δωρεάν για αναπτυξιακή χρήση (πχ Solaris) αλλά η υποστήριξη παρέχεται έναντι αμοιβής	Σχεδόν όλες οι δημοφιλής εκδόσεις παρέχονται δωρεάν. Η υποστήριξη επίσης παρέχεται δωρεάν μέσω διαφόρων φόρουμ όμως η εκτενείς υποστήριξη παρέχεται έναντι πληρωμής
Ασφάλεια	Μια πρόχειρη εκτίμηση των ιών που υπάρχουν για το UNIX είναι μεταξύ 85-120 οι οποίοι έχουν αναφερθεί μέχρι σήμερα. Εξαιρείτε το OS X το οποίο είναι ένα από τα χειρότερα λειτουργικά σε θέματα ασφάλειας σύμφωνα με έκθεση του 2015 που το κατατάσσει	Το Linux έχει περίπου 60-100 ιούς που έχουν αναφερθεί μέχρι σήμερα. Κανένας από αυτούς τους ιούς δεν εξαπλώνεται στις μέρες μας. Οι ευπάθειες βέβαια ποικίλουν από έκδοση σε έκδοση. Μία βασική αρχή είναι πώς οι δημοφιλέστερες εκδόσεις έχουν και τις περισσότερες

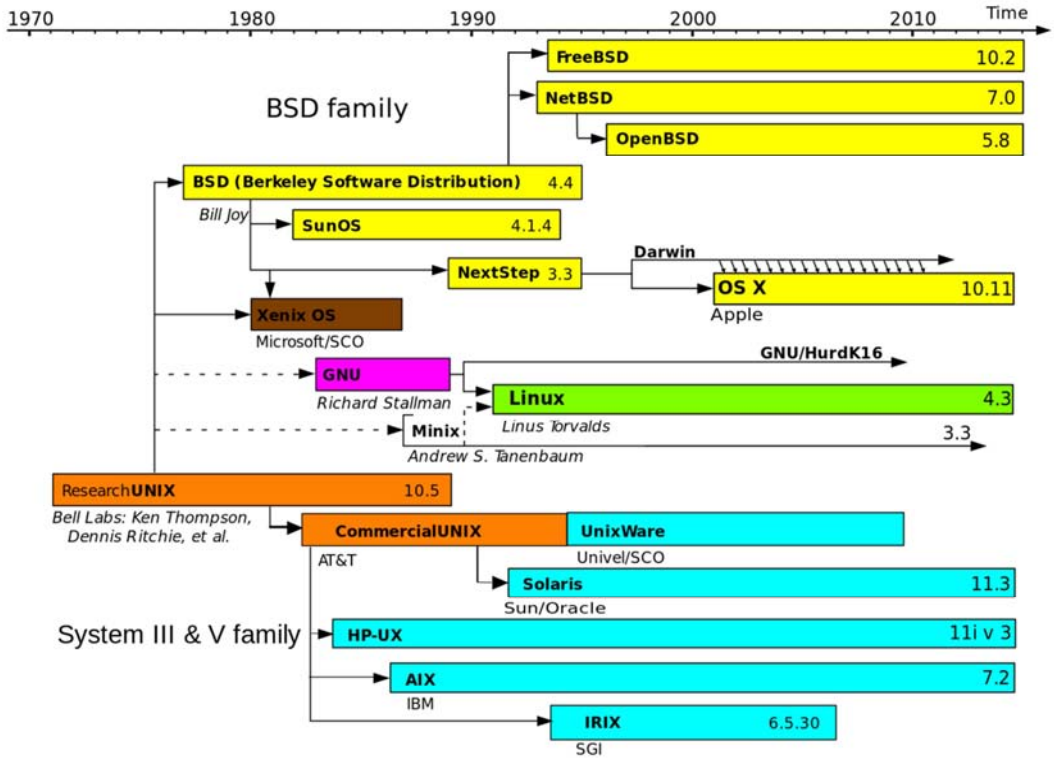
	στις πρώτες θέσεις με τις περισσότερες ευπάθειες	ευπάθειες
Ανίχνευση απειλών και λύσεις	Λόγω του ιδιοκτησιακού χαρακτήρα του αρχικού Unix, οι χρήστες πρέπει να περιμένουν αρκετά μέχρι να πάρουν τις σωστές διορθωμένες ενημερώσεις κώδικα για τις ευπάθειες που έχουν ανακαλυφθεί.	Στην περίπτωση του Linux, η ανίχνευση και η λύση στις απειλές που παρουσιάζονται είναι πολύ γρήγορη, καθώς το Linux συντηρείτε από τις διάφορες κοινότητες που έχουν αναλάβει την κάθε έκδοση. Έτσι όταν ανακαλύπτονται κάποιες ευπάθειες πολλοί προγραμματιστές αρχίζουν να εργάζονται πάνω σε αυτές από διάφορα μέρη του κόσμου με σκοπό την διόρθωσή τους
Αρχιτεκτονικές επεξεργαστών	x86/x64, Sparc, Power, Itanium, PA-RISC, PowerPC και πολλές άλλες.	Δεκάδες διαφορετικές
Παραδείγματα	OS X, Sun Solaris, IBM AIX, HP-UX Κλπ	Βασικές κύριες διανομές πάνω στις οποίες στηρίζονται ή έχουν στηριχθεί όλες οι υπόλοιπες: Slackware (SLS), Debian, Red Hat/Fedora, Arch, Enoch/Gentoo

Εκτός από τις παραπάνω επίσης διαφορές υπάρχουν και στους πυρήνες των δύο συστημάτων, στην υλοποίηση των συστημάτων, στην ιστορική περίοδο που άρχισε το καθένα να αναπτύσσεται, το λογισμικό/εφαρμογές που υπάρχουν και υποστηρίζονται και οι διάφορες άδειες. Για την ιστορία θα πρέπει να αναφέρουμε πώς το εγχείρημα GNU ανακοινώθηκε επίσημα το 1983 και τέθηκε σε λειτουργία από τον Ιανουάριο του 1984, με σκοπό τη δημιουργία ενός ολοκληρωμένου λειτουργικού συστήματος (GNU), τύπου Unix το οποίο θα ήταν ελεύθερο λογισμικό. Σήμερα, παραλλαγές του λειτουργικού GNU που χρησιμοποιούν τον πυρήνα του Linux, χρησιμοποιούνται ευρέως. Αν και αυτά τα συστήματα αναφέρονται συνήθως σαν "Linux", θα ήταν πιο ακριβές να αποκαλούνται συστήματα GNU/Linux.

Ο πυρήνας του Linux είναι παρόμοιος με αυτόν του Unix, αλλά δεν είναι ακριβώς το ίδιο. Ο πυρήνας σε συνδυασμό με τις εφαρμογές GNU, αποτελεί τη βάση για τις πολλές διανομές που γνωρίζουμε σήμερα που γενικά αναφέρεται ως «Linux». Κύρια διαφορά λοιπόν των συστημάτων Unix και Linux είναι πώς το πρώτο είναι κλειστού κώδικα λειτουργικό σύστημα ενώ το δεύτερο ανοιχτού. Αναλυτικότερα για το GNU/Linux στο κεφάλαιο 3.

[1]Η Open Group είναι μια παγκόσμια κοινοπραξία που επιτρέπει την επίτευξη των επιχειρηματικών στόχων μέσω προτύπων πληροφορικής. Με περισσότερους από 500 οργανισμούς-μέλη, έχει μια ποικιλόμορφη σύνθεση που εκτείνεται σε όλους τους τομείς της κοινωνίας της πληροφορικής - πελάτες, τα συστήματα και τους προμηθευτές λύσεων, πωλητές, συμβούλους, καθώς και ακαδημαϊκούς και ερευνητές.

Στο σχήμα 2.9 μπορούμε να παρατηρήσουμε την ιστορική εξέλιξη των συστημάτων που βασίζονται στο Unix ή ακόμη και εκείνα που είναι παρόμοια με αυτό. Γενικά όπως είπαμε και προηγουμένως μόνο τα συστήματα solaris, HP-UX, AIX, IRIX, UnixWare μπορούν να χαρακτηριστούν ως συστήματα Unix όλα τα υπόλοιπα, είτε έχουν παρόμοια δομή με το Unix και δημιουργήθηκαν με παρόμοιο τρόπο κάτω από άδειες ανοιχτού λογισμικού (Linux, Minix), είτε έχουν κάποιες κοινές συνδέσεις και κώδικα με το Unix αλλά διαφέρουν σε άλλα στοιχεία (OS X, BSD, Sun OS) δεν μπορούν να χαρακτηριστούν ως καθαρά συστήματα Unix. Για τις ανάγκες κατηγοριοποίησης της παρούσας διπλωματικής η ανάλυση του συστήματος GNU/Linux θα αναλυθούν στο κεφάλαιο 3 ενώ τα συστήματα BSD, OS X, Sun OS/Solaris θα αναφερθούν σε αυτό το κεφάλαιο κάτω από την ομπρέλα του Unix μιας και μοιάζουν περισσότερο με το Unix παρά με το Linux.



Σχήμα 2.9 Ιστορική εξέλιξη συστημάτων Unix

2.3.1 UnixWare OS

Το UnixWare είναι ένα λειτουργικό σύστημα που βασίζεται στο Unix. Αρχικά κυκλοφόρησε από την Univel, που από κοινού ανήκει στις εταιρίες AT&T Unix System Laboratories (USL) και στη Novell (. Στη συνέχεια εξαγοράστηκε από τη Novell. Αργότερα πήγε στην Caldera Systems, Caldera International, και στον όμιλο SCO πριν πωληθεί και πάλι στην UnXis (σημερινή Xinuos). Το UnixWare αναπτύσσεται ως επί το πλείστον σαν server. Δυαδικές διανομές του UnixWare είναι διαθέσιμες για υπολογιστές με αρχιτεκτονική x86. Το UnixWare είναι κατά κύριο λόγο εμπορικό ως λειτουργικό σύστημα server.

Ο πυρήνας του είναι μονολιθικός (κεφ. 1ο, παρ. 1.5.1.1) , και είναι κλειστού κώδικα που σημαίνει πώς είναι ένα λειτουργικό σύστημα που δεν πληροί τα κριτήρια για να είναι δωρεάν ή ανοικτού κώδικα. Οποιοδήποτε λογισμικό θέτει περιορισμούς στη χρήση, ανάλυση, τροποποίηση ή διανομή (την μεταβολή ή την τροποποίηση) μπορεί να χαρακτηριστεί ως ιδιόκτητο. Μια σχετική, αλλά διακριτή κατηγοριοποίηση στη βιομηχανία λογισμικού είναι το εμπορικό λογισμικό, το οποίο αναφέρεται σε λογισμικό που παράγεται προς πώληση, αλλά χωρίς απαραίτητα να σημαίνει ότι είναι κλειστού κώδικα.

Η τελευταία διανομή (distribution) του UnixWare είναι η έκδοση 7.1.4 (συλλογή στοιχείων: Μάρτιος 2016) σύμφωνα με την επίσημη σελίδα της εταιρείας (sco.com). Στον πίνακα 2.7 παρουσιάζεται το χρονοδιάγραμμα των εκδόσεων.

Year	Release	Company	Kernel Version
1991	UnixWare 1.0	Univel	1
1993	UnixWare 1.1	Novell	1
	UnixWare 1.1.1	Novell	1
	UnixWare 1.1.2	Novell	1
	UnixWare 1.1.3	Novell	1
1995	UnixWare 2.0	Novell	2.1
	UnixWare 1.1.4	Novell	1
1996	UnixWare 2.1	Santa Cruz Operat.	2.1
	UnixWare 2.1.1	Santa Cruz Operat.	2.1.1
	UnixWare 2.1.2	Santa Cruz Operat.	2.1.2
1998	UnixWare 2.1.3	Santa Cruz Operat.	2.1.3
1998	UnixWare 7	Santa Cruz Operat.	7.0.1
	UnixWare 7.0.1	Santa Cruz Operat.	7.0.1
1999	UnixWare 7.1.0	Santa Cruz Operat.	7.1.0
2000	UnixWare 7.1.1	Santa Cruz Operat.	7.1.1
2001	Open UNIX 8	Caldera International	7.1.2
2003	UnixWare 7.1.3	The SCO Group	7.1.3
2004	UnixWare 7.1.4	The SCO Group	7.1.4
2004	UnixWare 7.1.4 MP1	The SCO Group	7.1.4
2005	UnixWare 7.1.4 MP2	The SCO Group	7.1.4
2006	UnixWare 7.1.4 MP3	The SCO Group	7.1.4
2008	UnixWare 7.1.4 MP4	The SCO Group	7.1.4
2013-15	UnixWare 7.1.4+	Xinuos	7.1.4

Πίνακας 2.7 Χρονοδιάγραμμα εκδόσεων UnixWare (wikipedia.org)

2.3.2 Solaris OS

Το Solaris είναι ένα λειτουργικό σύστημα Unix που αρχικά αναπτύχθηκε από τη Sun Microsystems και αντικατέστησε το προηγούμενο λειτουργικό τους το Sun OS το 1993 (**το λειτουργικό σύστημα Solaris**, 2014). Μετά την εξαγορά της Sun από την Oracle Corporation το 2010, η τελευταία απέκτησε το λειτουργικό, το οποίο πλέον ονομάζετε Oracle Solaris. Το Solaris είναι γνωστό για τις δυνατότητες κλιμάκωσης/επεκτασιμότητας (scalability[1]), ειδικά για συστήματα SPARC, επειδή έχει ορισμένες ειδικές δυνατότητες όπως το Dtrace[2], το ZFS[3] και το Slider. Υποστηρίζει τις αρχιτεκτονικές SPARC, τους διακομιστές και σταθμούς εργασίας x86, από διάφορους κατασκευαστές. Είναι συμβατό με τις προδιαγραφές Single Unix Specification. Σαν μοντέλο, αποτελεί μίξη ανοικτού και κλειστού κώδικα. Ιστορικά, το Solaris αναπτύχθηκε σαν ιδιοταγές λογισμικό. Τον Ιούνιο του 2005 η Sun Microsystems διέθεσε μεγάλο μέρος του κώδικα κάτω από την άδεια CDDL, ξεκινώντας το σχέδιο OpenSolaris. Με το OpenSolaris, η Sun στόχευε στο να δημιουργήσει μια κοινότητα χρηστών γύρω από το λογισμικό. Μετά την απόκτηση της εταιρίας από την Oracle, τον Ιανουάριο 2010, η Oracle αποφάσισε να σταματήσει την ανάπτυξη του σχεδίου και την διανομή OpenSolaris, αλλά και το ανάλογο μοντέλο ανάπτυξης. Δέκα μέρες πριν, η είδηση διέρρευσε στους υπαλλήλους και ο Garrett D'Amore ανακοίνωσε το σχέδιο illumos. Τον Αύγουστο 2010, η Oracle σταμάτησε να παρέχει δημόσιες αναβαθμίσεις και τον πηγαίο κώδικα του πυρήνα Solaris, γυρνώντας πλέον το Solaris 11 σε ένα ιδιοταγές λειτουργικό σύστημα. Παρόλα αυτά, το Oracle Technology Network (OTN) και οι συνεργάτες της στην βιομηχανία λογισμικού μπορούσαν να είχαν πρόσβαση στην ανάπτυξη του πηγαίου κώδικα. Ο πηγαίος κώδικας για τα ανοικτού τύπου μέρη του λειτουργικού είναι διαθέσιμος από την Oracle.

Το Solaris χρησιμοποιεί μια κοινή βάση κώδικα για τις πλατφόρμες που υποστηρίζει, τις SPARC και τις x86 – x86-64. Έχει μια καλή φήμη πάνω στην συμμετρική πολυεπεξεργασία και υποστηρίζει μεγάλο αριθμό επεξεργαστών. Αυτό ιστορικά δένει με την αρχιτεκτονική SPARC της Sun και περιλαμβάνει υποστήριξη για τις 64bit εφαρμογές σε SPARC, από την έκδοση Solaris 7.

[1]Επεκτασιμότητα είναι η ικανότητα ενός συστήματος, δικτύου, ή μιας διαδικασίας να διαχειριστεί έναν αυξανόμενο όγκο εργασιών, ή την δυνατότητα του να διευρυνθεί, προκειμένου να ανταπεξέλθει στο συνεχές αυξανόμενο όγκο που του ανατίθεται.

[2]Το DTrace είναι ένα ολοκληρωμένο πλαίσιο δυναμικής ανίχνευσης που δημιουργήθηκε από την Sun Microsystems για την αντιμετώπιση προβλημάτων του πυρήνα και των εφαρμογών στα συστήματα παραγωγής σε πραγματικό χρόνο. Αρχικά αναπτύχθηκε για το Solaris, από τότε έχει κυκλοφορήσει κάτω από την άδεια Κοινής Ανάπτυξης και Διανομής (CDDL) και έχει κωδικοποιηθεί και για διάφορα άλλα Unix-like συστήματα.

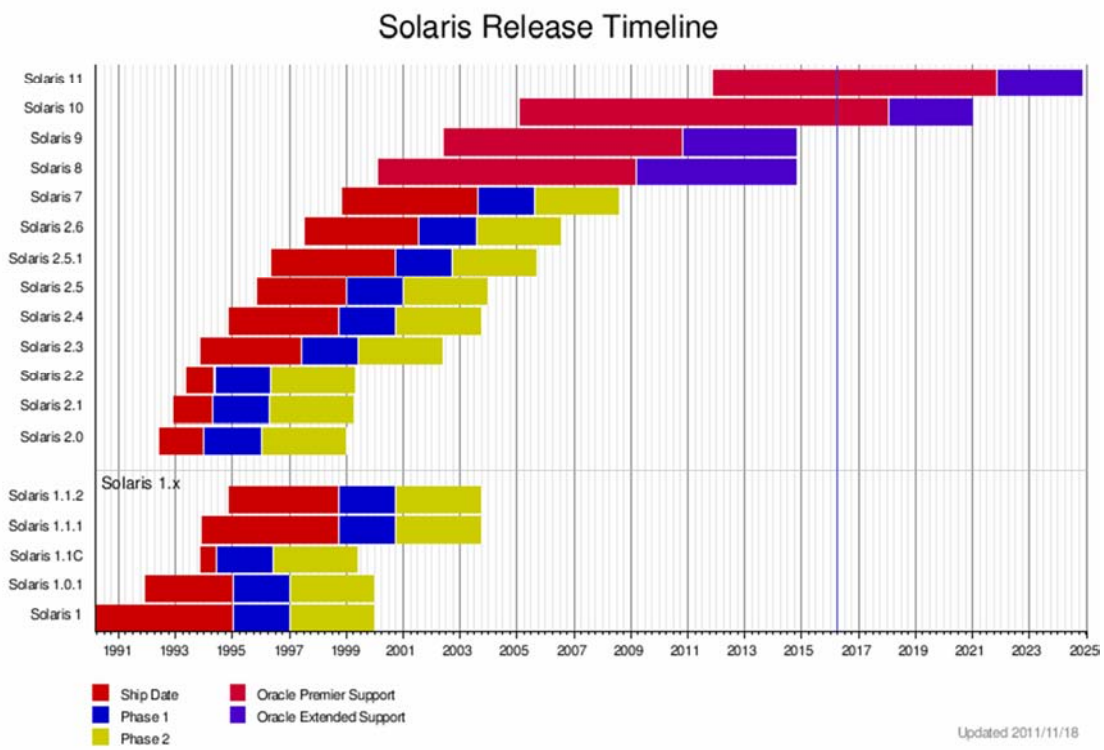
[3]Το ZFS είναι ένα συνδυασμένο σύστημα αρχείων και ένας διαχειριστής λογικών τόμων που σχεδιάστηκε από την Sun Microsystems. Τα χαρακτηριστικά του ZFS περιλαμβάνουν μέτρα προστασίας κατά της παραφθοράς δεδομένων, την υποστήριξη για μεγάλες δυνατότητες αποθήκευσης, την αποτελεσματική συμπίεση δεδομένων, την ενσωμάτωση των εννοιών του συστήματος αρχείων και τη διαχείριση του όγκου, συνεχή έλεγχο της ακεραιότητας και της αυτόματης επιδιόρθωσης, RAID-Z κλπ. Το όνομα ZFS έχει καταχωρηθεί ως σήμα κατατεθέν της Oracle Corporation

Μαζί με αυτές, διατέθηκε στην αγορά και ως ολοκληρωμένο πακέτο. Υποστήριξε τις αρχιτεκτονικές x86 από την έκδοση Solaris 2.1 και τις 64bit x86 εφαρμογές από την έκδοση Solaris 10, επιτρέποντας έτσι στη Sun να ηγηθεί στην αγορά 64bit επεξεργαστών, βασισμένων στην αρχιτεκτονική x86-64.

Η Sun έκανε σκληρό marketing στο Solaris για να χρησιμοποιείται πρωτίστως με τους δικούς της διακομιστές και σταθμούς εργασίας, βασισμένους σε AMD Opteron και Intel Xeon επεξεργαστές, τόσο, όσο και σε μηχανές που έφτιαχναν οι Dell, HP και IBM. Το 2009, οι παραπάνω κατασκευαστές υποστήριζαν το λειτουργικό, καθώς και η Fujitsu. Τον Ιούλιο του 2010, οι Dell και HP είπαν πως θα μεταπωλούν το Oracle Solaris, το Oracle Enterprise Linux και το Oracle VM και στις δικές τους πλατφόρμες.

Το Solaris μπορεί να εγκατασταθεί από διάφορες πηγές και πακέτα λογισμικού, που ποικίλλουν από μια μινιμαλιστική εγκατάσταση μέσω δικτύου, έως μια ολοκληρωμένη εγκατάσταση. Επιπρόσθετο λογισμικό περιλαμβάνει τον Apache, την MySQL, και άλλα. Μπορεί ακόμα να εγκατασταθεί από το sunfreeware και OpenCSW. Εγκατάσταση γίνεται από φυσικά μέσα ή από δίκτυο για χρήση σε εξυπηρετητές ή σε desktop.

Το χρονοδιάγραμμα των εκδόσεων του Solaris καθώς και η υποστήριξη αυτών παρουσιάζεται στο σχήμα 2.10



Σχήμα 2.10 Χρονοδιάγραμμα εκδόσεων του Solaris (πηγή: Wikipedia.org)

2.3.2.1 Το GUI στο Solaris

Οι πρώτες εκδόσεις του Solaris χρησιμοποιούσαν τα OpenWindows[1] σαν περιβάλλον εργασίας (**το λειτουργικό σύστημα Solaris**, 2014). Από τις εκδόσεις 2.0 έως 2.2 τα OpenWindows υποστήριζαν τα NeWS και τις εφαρμογές X, είχαν μάλιστα συμβατότητα για τις εφαρμογές SunView (από το παλιότερο περιβάλλον επιφανείας της Sun). Τα NeWS επέτρεπαν τη δημιουργία εφαρμογών σε PostScript[2], μια κοινή γλώσσα εκτύπωσης που δημιουργήθηκε στα 1982. Το σύστημα X Window ήρθε μέσα από το σχέδιο του MIT "Athena" (1984) και επέτρεπε σε εμφανιζόμενες εφαρμογές να αποσυνδέονται από το μηχάνημα το οποίο έτρεχαν, και βρισκόταν σε μια δικτυακή σύνδεση. Η σουίτα της Sun "SunView" έγινε στη συνέχεια port στα X. Η Sun αργότερα σταμάτησε την υποστήριξη για παλιές εφαρμογές SunView, και τα NeWS με τα OpenWindows 3.3, που ερχόντουσαν μαζί με το Solaris 2.3, και γύρισε στο X11R5 (ήταν η πέμπτη έκδοση του συστήματος παραθύρων X Windows, έκδοση 11 από το MIT που είχε κυκλοφορήσει τον Σεπτέμβριο του 1991) με την υποστήριξη display postscript[2]. Το γραφικό look and feel ήταν βασισμένο στο OpenLook.

Τα OpenWindows 3.6.2 ήρθαν στην τελευταία τους έκδοση με το Solaris 8. Οι εφαρμογές του OpenLook διαχειριστή παραθύρων σταμάτησαν να υποστηρίζονται στο Solaris 9, αλλά οι βιβλιοθήκες υποστήριξης συνέχισαν να υπάρχουν. Ο διαχειριστής παραθύρων OpenLook Virtual Window Manager (olvwm), μπορεί και σήμερα να γίνει download από το sunfreeware και δουλεύει σε εκδόσεις έως το Solaris 10. Η Sun και οι μεταπωλητές Unix δημιούργησαν στη συνέχεια μια συμμαχία στη βιομηχανία λογισμικού για να παράξουν τα standards στα Unix desktops. Σαν μέλος του COSE[3] (Common Open Software Environment), η Sun βοήθησε στην ανάπτυξη του CDE (Common Desktop Environment), σαν προσπάθεια να δημιουργηθεί ένα κοινό Unix περιβάλλον (GUI). Κάθε μεταπωλητής ανέλαβε διαφορετικά μέρη, η HP ανέλαβε τον διαχειριστή παραθύρων, η IBM έδωσε τον διαχειριστή αρχείων, και η Sun ανέπτυξε τις ευκολίες για e-mail και ημερολόγιο, όσο και την υποστήριξη για drug n' drop (Tool Talk). Αυτό το νέο περιβάλλον ήταν βασισμένο στο look and feel του Motif, και το Open Look θεωρήθηκε πεπαλαιωμένο. Το CDE πλέον ενοποίησε τα Unix desktops ανάμεσα σε πολλούς Open System μεταπωλητές. Ήταν διαθέσιμο σαν ένα ξεχωριστό πρόσθετο για το Solaris 2.4 και 2.5 και συμπεριλήφθηκε στις εκδόσεις του λειτουργικού από την 2.6 μέχρι την 10.

[1]Το OpenWindows ήταν ένα περιβάλλον εργασίας (GUI) για σταθμούς εργασίας της Sun Microsystems το οποίο συνδύαζε τα πρωτόκολλα SunView, NeWS, και X Window. Το OpenWindows περιλήφθηκε στην μεταγενέστερες εκδόσεις του SunOS 4 και του Solaris, μέχρι την απομάκρυνσή του, στην έκδοση Solaris 9 και την αντικατάστασή του από το Common Desktop Environment (CDE) και το GNOME 2.0.

[2]Το PostScript (PS) είναι μια γλώσσα για τη δημιουργία διανυσματικών γραφικών. Είναι μια δυναμικά γραφόμενη γλώσσα, αλυσιδωτού προγραμματισμού και δημιουργήθηκε από την Adobe Systems και τους John Warnock, Charles Geschke, Doug Brotz, Ed Taft και Μπιλ Πάξτον το 1982-4. Το Display PostScript (ή DPS) είναι ένα σύστημα 2D γραφικών που χρησιμοποιεί το μοντέλο PostScript (PS) απεικόνισης και γλώσσας (που αναπτύχθηκε αρχικά για την εκτύπωση) έτσι ώστε να δημιουργεί γραφικά στην οθόνη.

[3]Το Common Open Software Environment ή COSE ήταν μια πρωτοβουλία που έγινε το Μάρτιο του 1993 από τους μεγαλύτερους πωλητές Unix εκείνης της περιόδου για τη δημιουργία ανοιχτών προτύπων, και ενοποιημένων λειτουργικών συστημάτων (OS).

Η Sun στη συνέχεια έβγαλε μια προέκδοση με το ανοικτού κώδικα desktop GNOME 1.4 βασισμένο στην εργαλειοθήκη GTK+[1] στο Solaris 8. Το Solaris 9 παρουσιάστηκε με το GNOME 2.0 ως εναλλακτικό του CDE. Το Solaris 10 περιελάμβανε το Java Desktop System (JDS), σαν ένα κύριο συστατικό. Το JDS δεν συμπεριλήφθηκε στο Solaris 11, το οποίο ερχόταν με μια έκδοση του GNOME. Οι εφαρμογές CDE δεν υπάρχουν πλέον στο Solaris 11 αλλά υπάρχουν οι βιβλιοθήκες για να κρατούν την συμβατότητα.

Τα περιβάλλοντα ανοικτού κώδικα KDE και XFCE, αλλά και άλλοι διαχειριστές παραθύρων μπορούν να μεταγλωττιστούν και να τρέξουν στις τελευταίες εκδόσεις του Solaris. Η Sun είχε επενδύσει σε ένα νέο περιβάλλον επιφανείας που ονομαζόταν Project Looking Glass απο το 2003. Το σχέδιο όμως είναι ανενεργό από το 2006 και μετά.

2.3.2.2 Βασικά χαρακτηριστικά και στοιχεία του Solaris

- ✓ Ανάπτυξη: Oracle Corporation
- ✓ Γραμμένο σε: C, C++
- ✓ Λ/Σ: Unix (System V Release 4)
- ✓ Μοντέλο: Μείγμα κλειστού και ανοικτού κώδικα
- ✓ Πρώτη έκδοση: Ιούνιος 1992
- ✓ Τελευταία έκδοση: 11.3 / 26 Οκτωβρίου, 2015
- ✓ Στόχευση: Διακομιστές και Σταθμοί εργασίας
- ✓ Γλώσσα: Αγγλικά
- ✓ Αρχιτεκτονικές: SPARC, IA-32, x86-64, PowerPC (Solaris 2.5.1 μόνο)
- ✓ Kernel: Μονολιθικός Unix με δυναμικά loadable modules
- ✓ Περιβάλλον εργασίας: Java Desktop System / CDE / GNOME
- ✓ Άδειες: Διάφορες

[1]Η εργαλειοθήκη GIMP, (Gimp ToolKit) γνωστή σαν GTK+, είναι μια από τις δημοφιλέστερες εργαλειοθήκες widget για το X Window System για δημιουργία γραφικής διεπαφής χρήστη. Η GTK+ και η Qt έχουν αντικαταστήσει την Motif, που ήταν το πιο συχνά χρησιμοποιημένο X widget toolkit. Η GTK+ αναπτύχθηκε αρχικά για το πρόγραμμα επεξεργασίας γραφικών GNU Image Manipulation Program (GIMP), το 1997 από τους Spencer Kimball, Peter Mattis, και Josh MacDonald. Η άδεια διάθεσής του είναι η LGPL. Η GTK+ είναι ελεύθερο λογισμικό και αποτελεί μέρος του εγχειρήματος GNU.

[2]Το Java System Desktop (JSD), εν συντομία γνωστό ως OpenSolaris Desktop, είναι ένα γραφικό περιβάλλον που αναπτύχθηκε για πρώτη φορά από την Sun Microsystems και, στη συνέχεια, από την Oracle μετά την εξαγορά της sun από την Oracle το 2010. Το Java System Desktop είναι διαθέσιμο για το Solaris και κάποτε ήταν διαθέσιμα και για το Linux. Η έκδοση του Linux διεκόπη μετά την κυκλοφορία του Solaris ως λειτουργικό ανοικτού κώδικα το 2005. Το Java Desktop έχει ως στόχο να παρέχει ένα σύστημα γνώριμο στο μέσο χρήστη με μια πλήρη σουίτα λογισμικού γραφείου, όπως τα office, έναν web browser, e-mail, ημερολόγιο κλπ.

2.3.2.3 Το OpenSolaris και η μετεξέλιξή του σε OpenIndiana

Το OpenSolaris είναι ένα ανενεργό, ανοικτού κώδικα λειτουργικό σύστημα που βασίζεται στο Solaris και δημιουργήθηκε από την Sun Microsystems. Ήταν επίσης το όνομα του έργου που ξεκίνησε από τη Sun για την κατασκευή ενός έργου και κοινότητα χρηστών γύρω από το λογισμικό. Μετά την εξαγορά της Sun Microsystems το 2010, η Oracle αποφάσισε να διακόψει την ανάπτυξη του ανοιχτού λειτουργκού, και αντικατέστησε το μοντέλο διανομής OpenSolaris με το ιδιόκτητο Solaris Express.

Στις 14 Σεπτεμβρίου του 2010, το OpenIndiana ξεκίνησε επίσημα στο Κέντρο JISC στο Λονδίνο. Το OpenIndiana είναι η συνέχιση του OpenSolaris που σκοπό έχει να παραδώσει ένα λειτουργικό σύστημα της οικογένειας System V, το οποίο είναι binary-compatible με τα προϊόντα της Oracle Solaris 11 και Solaris 11 Express. Ωστόσο, αντί να βασίζεται γύρω από το OS/Net, το OpenIndiana έγινε μια διανομή με βάση το Illumos[1] (η πρώτη έκδοση εξακολουθεί να βασίζεται γύρω από το OS/Net). Το έργο χρησιμοποιεί το ίδιο σύστημα διαχείρισης πακέτων IPS[2] όπως και το OpenSolaris (**distrowatch**, 2016)

Πολλές άλλες διανομές συνυπάρχουν με το OpenIndiana (και βασίζονται στο OpenSolaris – Illumos) τα οποία όμως δεν κατέχουν τόση μεγάλη δημοτικότητα όσο το OpenIndiana.

2.3.3 HP-UX OS

Το HP-UX (από το Hewlett Packard Unix) είναι ιδιόκτητο λειτουργικό σύστημα της Hewlett Packard, βασισμένο στο UNIX System V (αρχικά System III) που κυκλοφόρησε για πρώτη φορά το 1984. Οι πρόσφατες εκδόσεις υποστηρίζουν την σειρά HP 9000, με βάση την αρχιτεκτονική PA-RISC, και τα συστήματα HP Integrity, με βάση την αρχιτεκτονική Itanium της Intel.

Το HP-UX ήταν το πρώτο Unix που προσέφερε λίστες ελέγχου πρόσβασης με δικαιώματα των αρχείου ως εναλλακτική λύση για το πρότυπο σύστημα αδειών Unix. Το HP-UX ήταν επίσης μεταξύ των πρώτων συστημάτων Unix που περιελάμβανε ένα ενσωματωμένο διαχειριστή λογικών τόμων. Η HP είχε μια μακρά συνεργασία με την εταιρία λογισμικού Veritas, και γι' αυτό χρησιμοποιεί το VxFS ως το κύριο σύστημα αρχείων.

Ο πυρήνας του είναι μονολιθικός και υποστηρίζει τα GUI: KDE, GNOME και CDE. Είναι ένα από τα πέντε εμπορικά λειτουργικά συστήματα που έχουν πιστοποιημένες εκδόσεις με το πρότυπο του UNIX 03 του ομίλου Open Group's.

[1]Το illumos είναι ένα δωρεάν και ανοικτού κώδικα λειτουργικό σύστημα Unix. Προέρχεται από το OpenSolaris, το οποίο με τη σειρά του προέρχεται από το SVR4 UNIX και το Berkeley Software (BSD). Το illumos περιλαμβάνει έναν πυρήνα, προγράμματα οδήγησης συσκευών, βιβλιοθήκες συστήματος και βοηθητικό λογισμικό για τη διαχείριση του συστήματος. Το illumos έχει παρόμοιο ρόλο με τον πυρήνα του Linux, ο οποίος αποτελεί τη βάση για τις διάφορες διανομές Linux. Έτσι, μετά την διάλυση του έργου ανοικτού κώδικα OpenSolaris υπάρχουν διάφορες διανομές ανοιχτού κώδικα του OpenSolaris που βασίζεται στον πυρήνα του illumos.

[2]Το Image Packaging System, επίσης γνωστό ως IPS ή PKG, είναι ένα cross-platform (γραμμένο σε Python) σύστημα διαχείρισης πακέτων που δημιουργήθηκε από την κοινότητα του OpenSolaris σε συνεργασία με τη Sun Microsystems.

2.3.4 AIX OS

Το AIX (Advanced Interactive eXecutive) είναι ένα ιδιόκτητο λειτουργικό σύστημα Unix που αναπτύχθηκε και πωλείται από την IBM για αρκετές από τις πλατφόρμες υπολογιστών της. Αρχικά κυκλοφόρησε για τον IBM 6150 RISC. Το AIX σήμερα υποστηρίζει ή έχει υποστηρίξει μια μεγάλη ποικιλία από πλατφόρμες υλικού, συμπεριλαμβανομένων των RS IBM/6000 series και αργότερα και τα συστήματα PowerPC-based, IBM System i, System/370, PS/2 προσωπικούς υπολογιστές, και το διακομιστή της Apple.

Το AIX βασίζεται στο UNIX System V με BSD συμβατές επεκτάσεις. Είναι και αυτό ένα από τα πέντε εμπορικά λειτουργικά συστήματα που έχουν πιστοποιηθεί με το πρότυπο του UNIX 03. Η οικογένεια AIX των λειτουργικών συστημάτων έκανε το ντεμπούτο της το 1986, και έγινε το πρότυπο λειτουργικό σύστημα για την σειρά RS/6000 κατά την έναρξή του το 1990, και εξακολουθεί να αναπτύσσεται ενεργά από την IBM. Σήμερα υποστηρίζει συστήματα τροφοδοσίας της IBM, IBMi και Linux. Το AIX ήταν το πρώτο λειτουργικό σύστημα που χρησιμοποίησε το σύστημα αρχείων journaling[1].

Το AIX κάνει χρήση ενός μονολιθικού πυρήνα και χρησιμοποιεί το κέλυφος Korn shell (ksh88). Το προεπιλεγμένο GUI είναι το CDE αλλά μπορούν να εγκατασταθούν και άλλα όπως το Plasma Workspaces και το GNOME. Σύμφωνα με την επίσημη ιστοσελίδα της IBM (ibm.com) το AIX βρίσκεται στην έκδοση 7.2 (συλλογή στοιχείων: Μάρτιος 2016) που κυκλοφόρησε στις 5 Οκτωβρίου 2015

2.3.5 IRIX OS

Το IRIX είναι ένα ανενεργό λειτουργικό σύστημα που αναπτύχθηκε από την Silicon Graphics (SGI) για να τρέχει σε σταθμούς εργασίας MIPS και servers. Βασίζεται στο UNIX System V με επεκτάσεις BSD. Το IRIX ήταν το πρώτο λειτουργικό σύστημα που συμπεριέλαβε το σύστημα αρχείων XFS. Η τελευταία σημαντική έκδοση του IRIX ήταν η 6.5 που κυκλοφόρησε τον Μάιο του 1998. Νέες μικρότερες εκδόσεις του IRIX 6.5 κυκλοφορούσαν κάθε τρίμηνο μέχρι το 2005. Από τότε έχουν υπάρξει τέσσερις ακόμη μικρές κυκλοφορίες. Μέσα από την έκδοση 6.5.22, υπήρχαν δύο κατευθύνσεις της κάθε έκδοσης: μία έκδοση συντήρησης (οι οποίες προσδιορίζονταν από το m επίθημα στον αριθμό έκδοσης) που περιελάμβανε μόνο διορθώσεις στον αρχικό κώδικα του IRIX 6.5, και μία άλλη έκδοση (με το επίθημα f), που περιελάμβανε βελτιώσεις και αναβαθμίσεις. Μια αναβάθμιση από την 6.5.x στην 6.5.22 έκδοση είναι διαθέσιμη ως δωρεάν download, ενώ οι εκδόσεις 6.5.23 και πάνω απαιτούν ένα ενεργό συμβόλαιο υποστήριξης της Silicon Graphics. Ο πυρήνας του είναι μονολιθικός και το GUI που χρησιμοποιεί είναι το IRIX Interactive Desktop.

[1]Το journaling είναι ένα σύστημα αρχείων που παρακολουθεί τις αλλαγές που δεν έχουν ακόμη δεσμευτεί στο κύριο μέρος του συστήματος αρχείων, καταγράφοντας τις προθέσεις αυτών των αλλαγών σε μια δομή δεδομένων γνωστή ως «journal», το οποίο είναι συνήθως ένα κυκλικό ημερολόγιο. Σε περίπτωση κατάρρευσης του συστήματος ή διακοπής ηλεκτρικού ρεύματος, τέτοια συστήματα αρχείων μπορούν να επανέλθουν πιο γρήγορα με μικρότερη πιθανότητα να καταστραφεί το περιεχόμενό τους

2.3.6 BSD OS

Το BSD (Berkeley Software Distribution μερικές φορές αναφέρεται και ώς Berkeley Unix) είναι ένα σύστημα προερχόμενο από το Unix, που διανέμεται από το πανεπιστήμιο του Μπέρκλεϊ της Καλιφόρνιας. Η ανάπτυξη του BSD ξεκίνησε τη δεκαετία του '70. Το όνομα χρησιμοποιείται επίσης για να περιγράψει το σύνολο των σύγχρονων διανομών που προέρχονται από τις αρχικές διανομές του BSD. Το BSD έτυχε ευρείας αναγνώρισης με τις εκδόσεις Unix που ήταν διαθέσιμες για "workstation-class" συστήματα. Αυτό μπορεί να αποδοθεί στην ευκολία με την οποία μπορούσε να χορηγηθεί άδεια για χρήση και την οικειότητα που είχαν με το σύστημα οι ιδρυτές πολλών τεχνολογικών επιχειρήσεων κατά τη δεκαετία του '80. Αυτή η οικειότητα κυρίως προήλθε από τη χρήση παρόμοιων συστημάτων κατά τη διάρκεια των σπουδών τους, όπως το Ultrix της DEC και το SunOS. Ενώ το ίδιο το BSD εκτοπίστηκε κατά ένα μεγάλο μέρος από τα συστήματα System V Release 4 και OSF/1 τη δεκαετία του '90 (από τα οποία και τα δύο ενσωμάτωσαν κώδικα του BSD), τα τελευταία χρόνια τροποποιημένες εκδόσεις ανοιχτού κώδικα του κυρίως κώδικα (προερχόμενες συνήθως από την έκδοση 4.4BSDlite) χρησιμοποιούνται και αναπτύσσονται συνεχώς.

Το 1995 εκδόθηκε η τελευταία έκδοση του 4.4BSD, η 4.4BSD-Lite Release 2. Μετά από αυτή το Computer Systems Research Group (CSRG) του Berkeley διαλύθηκε και η ανάπτυξη του BSD στο Berkeley σταμάτησε. Από τότε, διατηρούνται διάφορες διανομές βασισμένες στο 4.4BSD (όπως FreeBSD, OpenBSD, NetBSD, dragonfly κα), γνωστές ως BSD flavors (παραλλαγές του BSD). Για λόγους συντομίας αλλά και επειδή οι διάφορες εκδόσεις που κυκλοφορούν είναι παρόμοιες σε πολλά σημεία αν όχι σε όλα, θα ασχοληθούμε μόνο με την έκδοση FreeBSD και όχι με τις υπόλοιπες μιας και είναι η δημοφιλέστερη μεταξύ των χρηστών.

2.3.6.1 FreeBSD

Το FreeBSD είναι βασισμένο στο BSD UNIX που αναπτύχθηκε στο Πανεπιστήμιο του Berkeley στις δεκαετίες του 70 και 80. Το FreeBSD ξεκίνησε πρακτικά χρησιμοποιώντας ως αρχή το 4.4 BSD (και έπειτα το 4.4 BSD-Lite) το οποίο προσαρμόστηκε στους επεξεργαστές 386 στις αρχές της δεκαετίας του 1990 (**Παρουσίαση του FreeBSD,** 2013). Η αρχική προσαρμογή έγινε από τον William Jolitz (386BSD). 'Οταν ο William εγκατέλειψε την ανάπτυξη, άρχισε να δημιουργείται ένα ανεπίσημο patchkit (για να λειτουργεί το 386BSD σε ολοένα και πιο νέο hardware) το οποίο με την πάροδο του χρόνου φούσκωνε εκτός ελέγχου. Μια ομάδα προγραμματιστών αποφάσισε ότι το έργο άξιζε να συνεχιστεί οργανωμένα και έτσι γεννήθηκε το FreeBSD Project. Το FreeBSD χρησιμοποιείται σήμερα σε πολλούς διαφορετικούς τομείς: Ως εξυπηρετητής σε μεγάλα και μικρά sites (Yaboo, Netcraft, 5ony, DNS Root servers...), ως ερευνητικό λειτουργικό σε πανεπιστήμια και σχολεία, από ιδιώτες και επιχειρήσεις και αν θέλετε και ως desktop λειτουργικό. Στο FreeBSD βασίζονται επίσης διάφορες συσκευές (π.χ. NetApp routers) και μπορεί να χρησιμοποιηθεί εύκολα και ως embedded λειτουργικό (λόγω του μικρού μεγέθους).

Ένα σύστημα FreeBSD αποτελείται από:

- ✓ Το βασικό σύστημα — Πυρήνας και userland τα οποία παράγονται από την ίδια ομάδα και είναι πάντοτε συγχρονισμένα μεταξύ τους
- ✓ Τη Συλλογή των Ports (Ports Collection) με την οποία μπορούμε να εγκαταστήσουμε περισσότερες από 20000 πρόσθετες εφαρμογές. Οι περισσότερες ελεύθερες εφαρμογές μπορούν να μεταγλωττιστούν και να εκτελεστούν στο FreeBSD.

2.3.6.1.1 Βασικές Διαφορές από μια Διανομή GNU/Linux

Σύμφωνα με την ιστοσελίδα του FreeBSD και με τις παρουσιάσεις που έχουν γίνει, οι βασικές διαφορές απαρτίζονται στα παρακάτω στοιχεία:

- ✓ Ο πυρήνας και το userland του FreeBSD αναπτύσσονται από την ίδια ομάδα ανάπτυξης. Ο πηγαίος τους κώδικας αποθηκεύεται σε κοινό αποθετήριο (repository). Ένα σύστημα FreeBSD δεν συναρμολογείται από διάφορες εκδόσεις των βασικών προγραμμάτων, ο πυρήνας και το υπόλοιπο βασικό σύστημα είναι σε συγχρονισμό μεταξύ τους.
- ✓ Υπάρχει σαφής διάκριση μεταξύ του βασικού συστήματος και των υπόλοιπων εφαρμογών. Οι εφαρμογές εγκαθίστανται από έτοιμα πακέτα ή με τη βοήθεια του πηγαίου κώδικα (σύστημα Ports) αλλά είναι εντελώς διαχωρισμένες από το υπόλοιπο σύστημα. Μπορούμε πάντα να διαγράψουμε άμεσα όλες τις εφαρμογές και να μείνουμε με το βασικό σύστημα απείραχτο
- ✓ Η βασική άδεια χρήσης είναι το BSD License. Έχει λιγότερους περιορισμούς από την GPLκαι συχνά επίσης οδηγεί σε διαμάχες (license flames)
- ✓ Υπάρχουν και άλλες τεχνικές διαφορές, όμως κάποιος που μπορεί να χρησιμοποιήσει αποτελεσματικά ένα GNU/Linux σε περιβάλλον κονσόλας δεν Θα δυσκολευτεί ιδιαίτερα και με τοFreeBSD.

2.3.6.1.2 Ο πυρήνας (Kernel)

Ο πυρήνας του FreeBSD είναι ένας κλασικός μονολιθικός πυρήνας με δυνατότητες χρήσης modules (αρθρωμάτων). Καθώς ο πηγαίος κώδικας και ο μεταγλωττιστής συνοδεύει το σύστημα (**Παρουσίαση του FreeBSD,** 2013), είναι αρκετά εύκολο να δημιουργήσουμε τον δικό μας προσαρμοσμένο πυρήνα προσθέτοντας ήαφαιρώντας δυνατότητες και συσκευές. Ο πυρήνας του FreeBSD είναι γνωστός για τη σταθερή και αξιόπιστη λειτουργία του ακόμα και κάτω από συνθήκες μεγάλου φόρτου. Η απόδοση του σε συστήματα πολυεπεξεργαστών έχει ανέβει αισθητά από την έκδοση 7.0 και μετά. Υποστηρίζει μεγάλο πλήθος συσκευών — αν και φυσικά μικρότερο από τον πυρήνα του Linux.

2.3.6.1.3 Εγκατάσταση Εφαρμογών

Είναι αρκετά εύκολο να εγκαταστήσουμε εφαρμογές σε ένα σύστημα FreeBSD. Για το σκοπό αυτόμπορούμε να χρησιμοποιήσουμε είτε:

- Έτοιμα πακέτα. Τα πακέτα εγκαθίστανται με πολύ απλό τρόπο και αρκετά γρήγορα. Ένα πακέτο το οποίο χρειάζεται και άλλα για να εγκατασταθεί (εξαρτήσεις) θα τα ανακτήσει και θα τα εγκαταστήσει επίσης αυτόματα.
- Μέσω του πηγαίου κώδικα και της συλλογής των Ports. Η συλλογή των Ports είναι ένα αυτόματο σύστημα ανάκτησης, προσαρμογής, μεταγλώττισης και εγκατάστασηςπρογραμμάτων από τον πηγαίο κώδικα. Όπως και με τα πακέτα, η εγκατάσταση ενόςπρογράμματος από τα Ports εγκαθιστά αυτόματα και όλες τις εξαρτήσεις. Το πλεονέκτημα των Ports είναι ότι μπορούμε να προσαρμόσουμε το πρόγραμμα (αλλάζοντας διάφορες επιλογές) κατά τη μεταγλώττιση. Συχνά τα Ports έχουν πιο νέες εκδόσεις από τα έτοιμα πακέτα. Τα μεγάλα Ports θα χρειαστούν ωστόσο πολύωρη μεταγλώττιση.

Άσχετα με την έκδοση του βασικού συστήματος που έχουμε (εφόσον δεν είναι υπερβολικά παλιό), μπορούμε πάντα να χρησιμοποιήσουμε τις τελευταίες εκδόσεις των εφαρμογών. (**Παρουσίαση του FreeBSD,** 2013)

2.3.6.1.4 Γραφικά Περιβάλλοντα (GUI)

Από προεπιλογή, το FreeBSD δεν έχει κανένα γραφικό περιβάλλον. Μπορούμε όμως να εγκαταστήσουμε και να ρυθμίσουμε το περιβάλλον της αρεσκείας μας. Το FreeBSD χρησιμοποιεί τη διανομή Xorg του X11, και μπορούμε να εγκαταστήσουμε τα παρακάτω γραφικά περιβάλλοντα:

- ✓ GNOME
- ✓ KDE
- ✓ Η οποιοδήποτε άλλο από τα περίπου 150 "μικρά" γραφικά περιβάλλοντα που διατίθενταιστη συλλογή των Ports (**Παρουσίαση του FreeBSD,** 2013)

2.3.6.1.5 Διαχείριση Συστήματος

- ✓ Το FreeB5D χρησιμοποιεί το σύστημα εκκίνησης "rc". Οι περισσότερες (αν και όχι όλες) διανομές Linux χρησιμοποιούν το SySV σύστημα. Το rc σύστημα είναι αρκετά απλό.
- ✓ Το FreeBSD δεν διαθέτει runlevels[1]. Υπάρχουν μόνο δύο καταστάσεις λειτουργίας, singleuser και full multiuser mode.

[1]Το Runlevel είναι ένας τρόπος λειτουργίας σε ένα λειτουργικό σύστημα που εφαρμόζει το στυλ του Unix System V. Υπάρχουν επτά runlevels, αριθμημένα από το μηδέν έως το έξι (αν και μπορεί να χρησιμοποιηθούν μέχρι και δέκα 0-9).

- ✓ Το βασικό σύστημα αρχείων του FreeBSD είναι το UFS2. Μπορεί να συνδυαστεί και με λειτουργίες journalling. Υποστηρίζονται και άλλα συστήματα όπως το ZFS.
- ✓ Το FreeBSD υποστηρίζει εικονικές κονσόλες με τη χρήση των πλήκτρων ALT+F1, F2 κλπ.
- ✓ Υπάρχουν τρία διαθέσιμα firewalls για να διαλέξουμε
- ✓ Προχωρημένες λειτουργίες ασφαλείας: MAC (Mandatory Access Control), Auditing, Jailsκαι Kernel Securelevels[1].
- ✓ Διάφορες λειτουργίες RAID[2]: Mirroring, Striping κλπ.

2.3.6.1.6 Το FreeBSD ως Server

'Όλες οι γνωστές εφαρμογές που χρειάζονται για τη δημιουργία ενός τοπικού ή Internet serverπεριέχονται στο βασικό σύστημα και στη συλλογή των Ports:

- ✓ Apache, MySQL και PHP για την ανάπτυξη και υποστήριξη δυναμικών sites
- ✓ Sendmail MTA στο βασικό σύστημα και πλήθος άλλων MTA στη συλλογή των Ports. Π.χ.Postfix, exim. Αντίστοιχα υπάρχει και πλήθος POP3 και IMAP servers.
- ✓ NFS (Network Filesystem), NIS (Network Information System), BIND (DNS Server)υπάρχουν στο βασικό σύστημα.
- ✓ Ο πυρήνας του FreeBSD υποστηρίζει μεγάλο πλήθος δικτυακού υλικού (π.χ. κάρτεςδικτύου). Καθώς το FreeBSD χρησιμοποιείται πολύ συχνά σε μεγάλους εξυπηρετητές,υποστηρίζεται και "εξωτικό" υλικό (κάρτες δικτύου 10 Gbit, fiber κλπ).

2.3.6.1.7 Το UbuntuBSD και άλλες παραλλαγές συστημάτων

Το τελευταίο χρονικό διάστημα αρκετές παραλλαγές συστημάτων έχουν προκύψει. Σημαντική παραλλαγή του BSD θα μπορούσε να θεωρηθεί το UbuntuBSD. Πρόκειται για ένα λειτουργικό σύστημα με χαρακτηριστικά Ubuntu αλλά προσαρμοσμένο με τον πυρήνα του BSD (**iguru.gr**). Η διπλωματική δεν θα επεκταθεί περισσότερο στις συγκεκριμένες παραλλαγές καθώς και σε πολλές άλλες μειονοτικές εκδόσεις και διανομές διότι θα ήταν αδύνατο να αναλυθούν και να αναφερθούν όλες.

[1]Το Securelevel είναι ένας μηχανισμός ασφάλειας στον πυρήνα του BSD, ο οποίος μπορεί προαιρετικά να περιορίσει ορισμένες δυνατότητες. Το Securelevel ελέγχεται από μία μεταβλητή που ονομάζεται sysctl (kern.securelevel). Αυτή η τιμή είναι ένας ακέραιος, η οποία ρυθμίζεται σε μια τιμή > 0 που επιτρέπει ορισμένες κατηγορίες περιορισμών. Κάθε διαδικασία superuser μπορεί να αυξήσει την Securelevel, αλλά μόνο η διεργασία init μπορεί να τη μειώσει.

[2]Το RAID είναι μια τεχνολογία εικονικοποίησης αποθήκευσης δεδομένων που συνδυάζει πολλαπλές φυσικές μονάδες δίσκων σε μια ενιαία λογική μονάδα, για τους σκοπούς του πλεονασμού δεδομένων, τη βελτίωση των επιδόσεων, ή και τα δύο.

2.3.7 OS X

Το OS X, πρώην Mac OS X είναι μια σειρά γραφικών λειτουργικών συστημάτων που αναπτύσσεται, προωθείται και πωλείται από την Apple Inc. και συμπεριλαμβάνεται σε κάθε καινούριο υπολογιστή Macintosh (Mac). Το Mac OS X είναι η εξέλιξη του αρχικού Mac OS το οποίο ήταν το αρχικό λειτουργικό σύστημα της Apple την περίοδο 1984-1999. Σε αντίθεση με το αρχικό Mac OS, το Mac OS X είναι ένα UNIX λειτουργικό σύστημα το οποίο άρχισε να αναπτύσσεται στην εταιρία NeXT από τα τέλη του 1980 και μέχρι την εξαγορά της από την Apple το 1997. Η πρώτη έκδοσή του κυκλοφόρησε το 1999 ως Mac OS X Server 1.0, ενώ η πρώτη έκδοσή του για επιτραπέζιους και φορητούς υπολογιστές, Mac OS X v10.0 "Cheetah", ακολούθησε τον Μάρτιο του 2001. Ο πυρήνας που χρησιμοποιείτε είναι υβριδικός (κεφ. 1^{ο}, παραγ. 1.5.1) και η τελευταία σταθερή έκδοση του λειτουργικού είναι η 10.9.4 (Mavericks) που κυκλοφόρησε στις 30 Ιουλίου 2014. Το GUI που χρησιμοποιείτε είναι το Aqua το οποίο είναι και αυτό δημιουργία της Apple. Να τονιστεί πώς είναι και αυτό ένα λειτουργικό σύστημα κλειστού τύπου. Πολλά θέματα ασφαλείας έχουν παρουσιαστεί το τελευταίο χρονικό διάστημα κατατάσσοντας το λειτουργικό σύστημα OS X στα χειρότερα σε θέματα ασφαλείας και ευπαθειών (**cvedetails.com**, 2015)

Κεφάλαιο 3

3.1 Αναφορά στην ιστορία του Linux και των διανομών (distributions)

Η ιστορία του Linux λίγο πολύ γνωστή με τον Linus Torvalds φοιτητή τότε (1991) στο Πανεπιστήμιο του Ελσίνκι, να σχεδιάζει εκ του μηδενός ένα σύστημα κοντά στα πρότυπα του Unix και του Minix. Η ανάπτυξη του Linux επηρεάστηκε δε από το λειτουργικό του Unix όμως η συγγραφή του άρχισε από το μηδέν, δηλαδή έχουμε να κάνουμε με ένα λειτουργικό σύστημα που δεν έχει πάρει καθόλου έτοιμο κώδικα από το Unix. Βέβαια καθώς ο Linus μέχρι τότε χρησιμοποιούσε το Unix και το Minix βλέπουμε να έχει δανειστή αρκετά στοιχεία και από τα δύο συστήματα. Στην αρχή η άδεια χρήσης που χρησιμοποιούσε ο Torvalds δεν επέτρεπε την αναδιανομή για εμπορικούς σκοπούς. Αργότερα το έκανε διαθέσιμο υπό την άδεια GNU GPL. Έτσι το GNU βρήκε έναν πυρήνα για να λειτουργήσει, και το Linux βρήκε έτοιμη μια μεγάλη ποικιλία προγραμμάτων. Εντάσσοντας το εγχείρημά του στο GNU, η ανάπτυξη του Linux ήταν αλματώδης και γρήγορα ξεπέρασε το MINIX σε δημοτικότητα. Χιλιάδες προγραμματιστές από τότε μέχρι και σήμερα έχουν συνεισφέρει στην ανάπτυξη του πυρήνα του Linux αλλά και άλλων στοιχείων που απαρτίζουν το λειτουργικό σύστημα γενικά. Ας αναφερθούμε όμως σε μερικά τεχνικά στοιχεία τα οποία δεν είναι ιδιαίτερα γνωστά για τα πρώτα συστήματα του Linux και τις διανομές.

Στα τέλη του 1991, όταν το Linux έγινε διαθέσιμο στο internet, δεν υπήρχαν οι διάφορες διανομές (distributions) όπως συμβαίνει σήμερα. Υπήρχαν δισκέτες 5.25΄΄ που μπορούσαν να χρησιμοποιηθούν για να τρέξει ένα σύστημα Linux. Η εκκίνηση γινόταν από μια δισκέτα εκκίνησης και στη συνέχεια, όταν ζητούνταν από το σύστημα, έπρεπε να τοποθετηθεί η ριζική δισκέτα. Μετά από λίγο είχαμε διαθέσιμη μία γραμμή εντολών που μπορούσαμε να εισάγουμε τις εντολές μας και να αλληλεπιδράσουμε με το σύστημά μας. Εκείνες τις ημέρες, αν ήθελε κανείς να κάνει εκκίνηση από το σκληρό δίσκο θα έπρεπε να χρησιμοποιήσει ένα σημειωματάριο hex στην κύρια εγγραφή εκκίνησης του δίσκου προκειμένου να δημιουργήσει ένα script εκκίνησης. Κάτι που ήταν σίγουρα αρκετά χρονοβόρο και δύσκολο στην υλοποίηση για τους απλούς χρήστες. Οι λεγόμενες διανομές του Linux δημιουργήθηκαν και έγιναν διαθέσιμες πολύ αργότερα. Κάποιες από τις πρώτες διανομές Linux ήταν οι MCC interim (Φεβρουάριος 1992), TAMU, Yggdrasil (Δεκέμβριος 1992 – ήταν επίσης η πρώτη εταιρία που έκανε χρήση ενός live-cd), SLS (Μάιος 1992) κλπ. Σημαντικότερη από αυτές η SLS από την οποία προήλθε η παλαιότερη εν χρήση σήμερα διανομή το Slackware Linux με ιδρυτή τον Patrick Volkerding. Από την ίδια διανομή επηρεάστηκε και το Debian Linux του Ian Murdock. Αν και η SLS ήταν μία διανομή πολύ δημοφιλής, τα πολλά bugs που είχε δεν τις επέτρεψαν την συνέχισή της, με αποτέλεσμα την απαξίωση της και την αντικατάσταση από άλλες διανομές. Για την ιστορία είναι επιβεβλημένο να κάνουμε ειδική μνεία στην διανομή MCC interim η οποία κατέχει και τον τίτλο της πρώτης διανομής Linux.

3.1.1 Η πρώτη διανομή Linux (MCC interim)

Σύμφωνα με τα ιστορικά στοιχεία των Read me αρχείων της έκδοσης, σκοπός της ήταν να επιτρέψει σε άτομα τα οποία δεν ήταν εξειδικευμένοι χρήστες Unix να εγκαταστήσουν μία έκδοση του λειτουργικού συστήματος Linux στον υπολογιστή τους. Η εγκατάσταση θα γινόταν αυτόνομα και θα ήταν εύκολα επεκτάσιμη.
Όταν η έκδοση 0.10 του Linux εμφανίστηκε για πρώτη φορά το φθινόπωρο του 1991, εκδόθηκε από τον Linus Torvalds ως ένα ζευγάρι δισκέτες, που είναι γνωστές ως η "Boot" και "root" δισκέτες. Ο ίδιος ο Linus κυκλοφόρησε πολλά βοηθητικά προγράμματα τα οποία όμως δεν χωρούσαν σε αυτές τις δισκέτες. Τελικά ένας αρκετά μεγάλος αριθμός βοηθητικών προγραμμάτων ήταν διαθέσιμα. Οι διανομές λογισμικού ήταν χαοτικές, και χρειάζονταν πολλή προσπάθεια για να συλλέξει κανείς όλα εκείνα που χρειάζονταν για να λειτουργήσει ένα πλήρη σύστημα.

Το Manchester Computing Centre (MCC - από αυτό προκύπτει και το όνομα της διανομής) είναι ένα τμήμα του Πανεπιστημίου του Manchester που υποστήριζε τους υπολογιστές του Πανεπιστημίου του MIT και άλλων Πανεπιστημίων. Έτσι το Νοέμβριο του 1991 έκαναν διαθέσιμο το Linux μέσω ftp και τον Φεβρουάριο του 1992 έκαναν διαθέσιμη την διανομή MCC interim με σκοπό την συγκέντρωση όλων των στοιχείων και προγραμμάτων που ήταν διαθέσιμα και συνιστούσαν ένα λειτουργικό σύστημα Linux.

Η MCC interim κάνει χρήση του κωδικού ramdisk ο οποίος γράφτηκε από τον Theodore Ts'o με σκοπό να προσαρτά έναν εικονικό δίσκο ρίζας κατά την εκκίνηση από τον πυρήνα της δισκέτας. (Αυτό το είδος της διανομής είχε χρησιμοποιηθεί νωρίτερα από τον Andrew Tannenbaum για να διανείμει το λειτουργικό του σύστημα MINIX). Με την έκδοση 0.99p8+, εγκαταλείφθηκε η ramdisk-as-root και υιοθετήθηκε μια συνδυασμένη δισκέτα ρίζας εκκίνησης (bootable root disk). Το πρώτο bootable root disk δημιουργήθηκε από τον H J Lu (Ο οποίος έκανε ουσιαστική χρήση του λογισμικού Werner Almesberger LILO – το οποίο είναι διαθέσιμο ακόμη και σήμερα ως εκκινητής με διάφορες βελτιώσεις, αν και υπάρχουν πολλοί πιο εξελιγμένοι boot loaders προς χρήση). Σε αντίθεση με την bootable root disk του H J, ο δίσκος MCC δεν περιέχει ένα εκτελέσιμο κέλυφος. Μετά την έκδοση 1.0+, το ελάχιστο βασικό σύστημα δεν χωρούσε πλέον σε μία μόνο δισκέτα και έτσι υιοθετήθηκε η χρήση δύο δισκετών, που όμως καμία δεν περιείχε ένα εκτελέσιμο κέλυφος.

Όπως είπαμε και προηγουμένως μετά την έκδοση του MCC interim πολλοί ήταν εκείνοι που διένειμαν και άλλες διανομές παρόμοιες με την MCC interim.

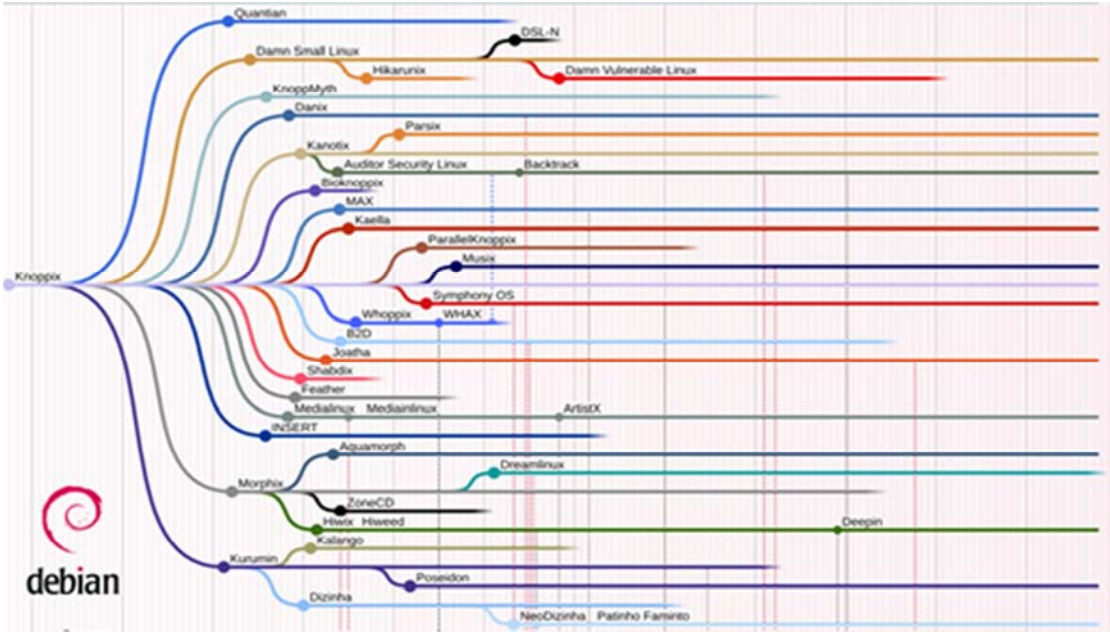
3.1.2 Οι σημαντικότερες διανομές και η εξελικτική τους πορεία

Σύμφωνα με το χρονικό των διανομών, οι πέντε κύριες διανομές Linux πάνω στις οποίες βασίζονται όλες οι υπόλοιπες σήμερα είναι οι:

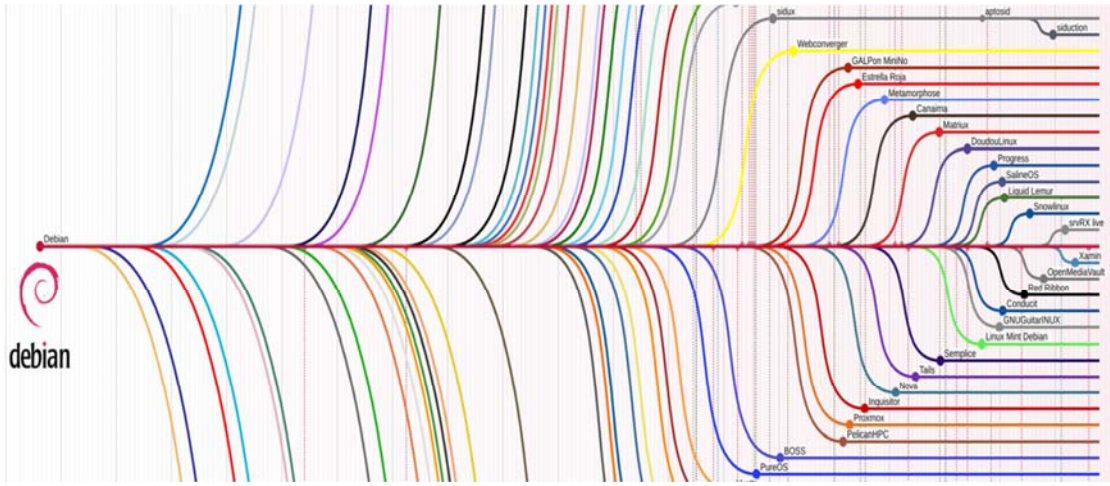
1. Debian
2. Slackware
3. Red hat
4. Arch
5. Enoch/Gentoo

3.1.2.1 Η διανομή Debian

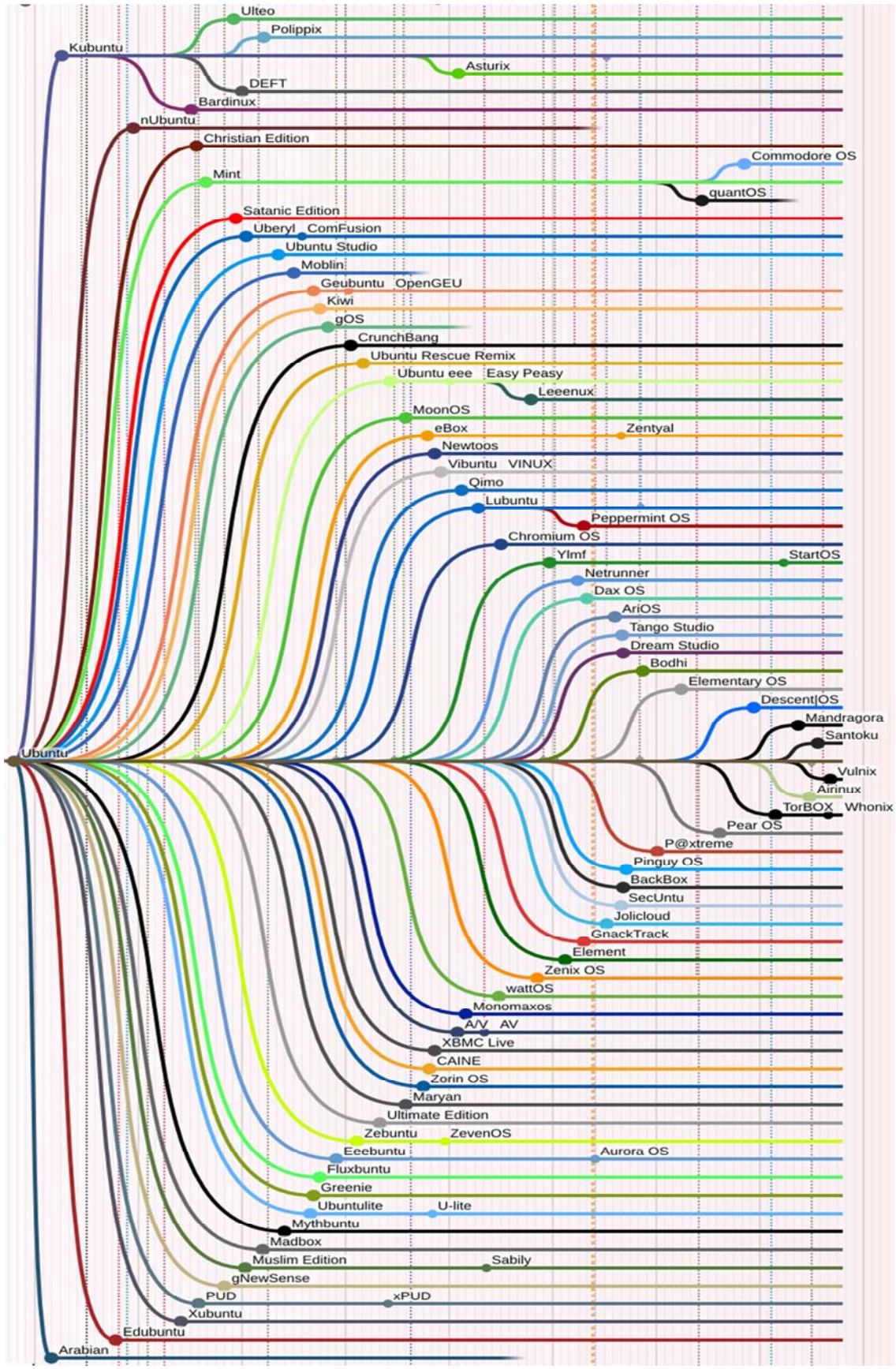
Η διανομή Debian είναι εκείνη που κρατά τα ινία της ποιο επιτυχημένης διανομής από άποψη επηρεασμού των σύγχρονων διανομών που βασίζονται πάνω σε αυτή. Με πάνω από 180 διανομές βασιζόμενες στο Debian, συμπεριλαμβανομένων των δημοφιλών διανομών Ubuntu, Linux Mint και knoppix, το Debian δεν θα μπορούσε παρά να θεωρηθεί ως η κατεξοχήν διανομή που ίσως να χρησιμοποιούν οι περισσότεροι οι οποίοι εισέρχονται στον χώρο του Linux. Αυτό μαζί με το γεγονός ότι οι εκδόσεις που βασίζονται πάνω στο Debian έχουν φιλικότητα ως προς την χρήση, την εγκατάσταση του συστήματος και των διαχειριστή πακέτων, κατατάσσει την διανομή αυτή ως μία ελκυστική επιλογή εύκολη προς χρήση. Στα σχήματα 3.1, 3.2, 3.3 παρουσιάζονται μερικές από τις διανομές που προέρχονται από την διανομή Debian.



Σχήμα 3.1 Διανομές Debian – Knoppix (πηγή: futurist.se/gldt/)



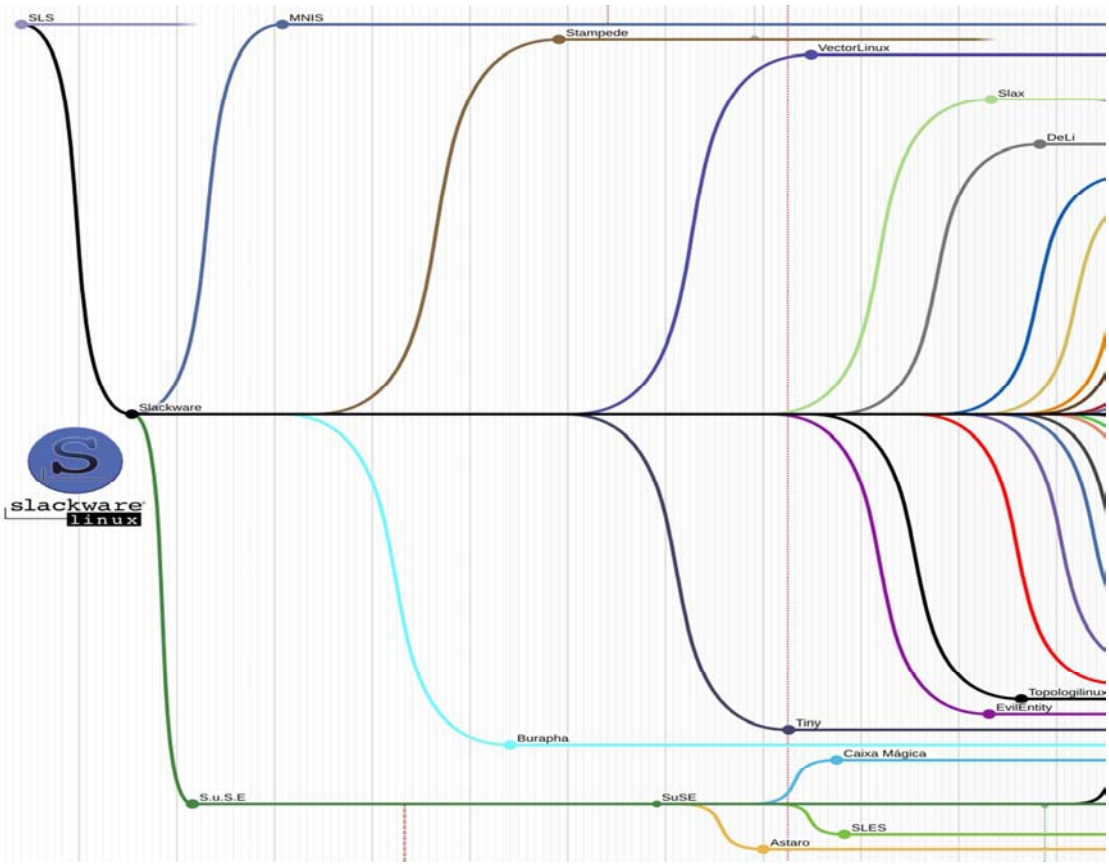
Σχήμα 3.2 Διανομές Debian (πηγή: futurist.se/gldt/)



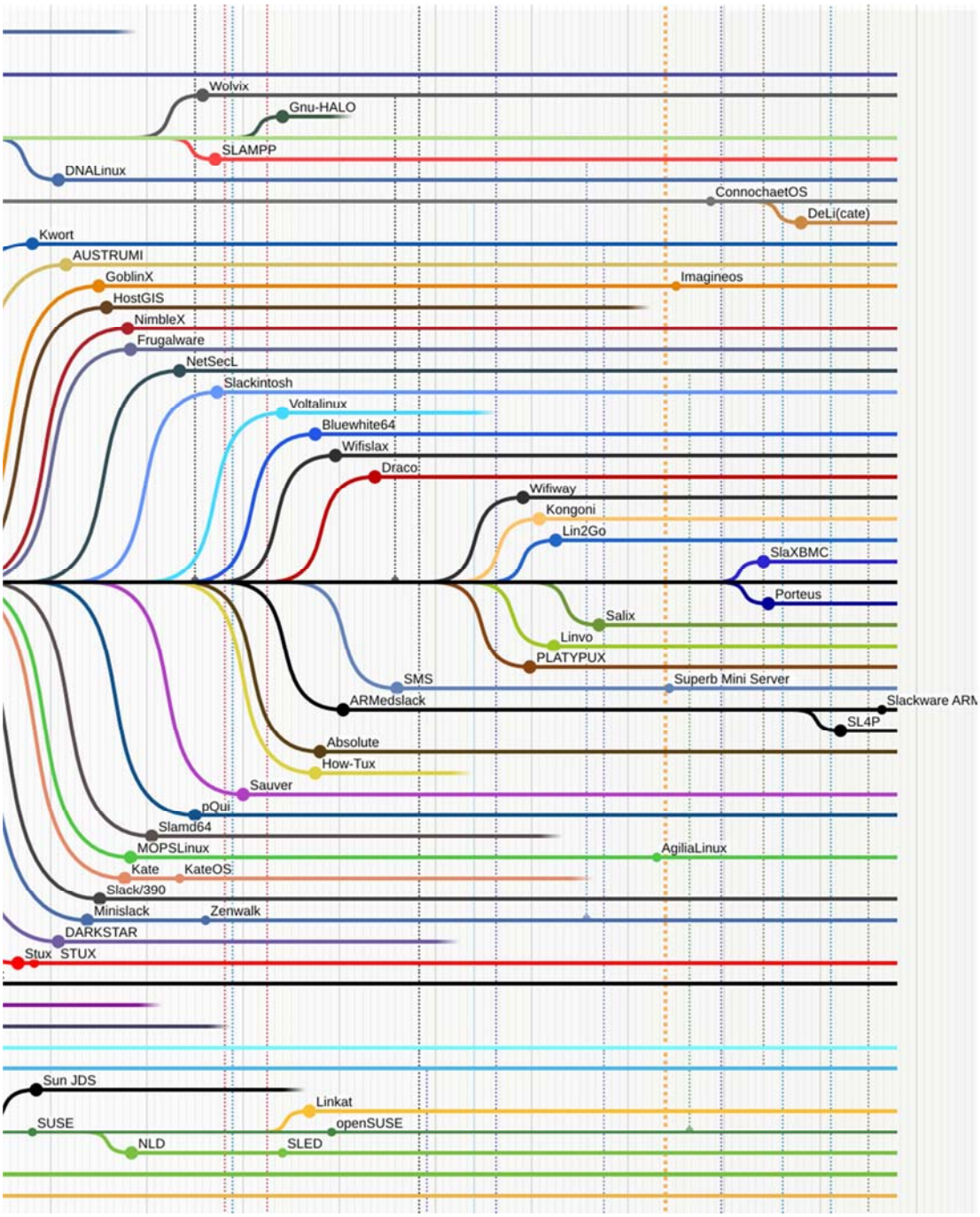
Σχήμα 3.3 Διανομές Debian-Ubuntu (πηγή: futurist.se/gldt/)

3.1.2.2 Η διανομή Slackware

Το Slackware ως η παλαιότερη εν ενεργεία διανομή που έως σήμερα συντηρείται διεκδικεί επάξια την θέση του πιο σταθερού συστήματος, ενώ από αυτό έχουν προέλθει οι δημοφιλής διανομές SUSE, OpenSUSE και Slax. Ένα ρητό λέει πώς εκείνος που ξέρει κάποια έκδοση του Linux απλά γνωρίζει την συγκεκριμένη έκδοση και τίποτα περισσότερο αλλά εκείνος που γνωρίζει το Slackware ξέρει το Linux σε βάθος. Το ρητό αυτό δεν είναι τυχαίο καθώς το Slackware περιλαμβάνει στα αποθετήριά του πακέτα τα οποία σχετίζονται με τον πυρήνα του συστήματος απευθείας. Ως εκ τούτου, εκτός από τα βασικά πακέτα, θα πρέπει να βρούμε το λογισμικό που θέλουμε μόνοι μας και να το εγκαταστήσουμε. Αυτό προϋποθέτει ότι η δημιουργία των πακέτων Slackware θα πρέπει να γίνει από τον χρήστη, χρησιμοποιώντας κάποια εργαλεία για μετατροπή των αρχείων .rpm και .deb, ή κάνοντας compiled των κώδικα μόνοι μας. Μπορούμε επίσης να εγκαταστήσουμε τα πακέτα του Slackware με την εντολή/εργαλείο pkgtool, αλλά αυτό το εργαλείο δεν κάνει τίποτα περισσότερο από το να εγκαταστήσει ένα πακέτο και να το παρακολουθεί - δεν κάνει καμία ανάλυση των εξαρτήσεων που μπορεί να χρειάζεται το πρόγραμμα προκειμένου να λειτουργήσει σωστά, οπότε ο χρήστης θα πρέπει να είναι γνώστης των εξαρτήσεων κάτι το οποίο είναι αρκετά δύσκολο και χρονοβόρο. Η διανομή Slackware θα αναλυθεί περισσότερο σε επόμενη ενότητα καθώς παρουσιάζει ιδιαιτερότητες και σημεία τα οποία έχουν ενδιαφέρον για περαιτέρω έρευνα. Στα σχήματα 3.4.1 και 3.4.2 βλέπουμε μερικές από τις διανομές που προήλθαν από την διανομή του Slackware η οποία προήλθε και αυτή από την διανομή SLS.



Σχήμα 3.4.1 Διανομές Slackware και SuSE (πηγή: futurist.se/gldt/)

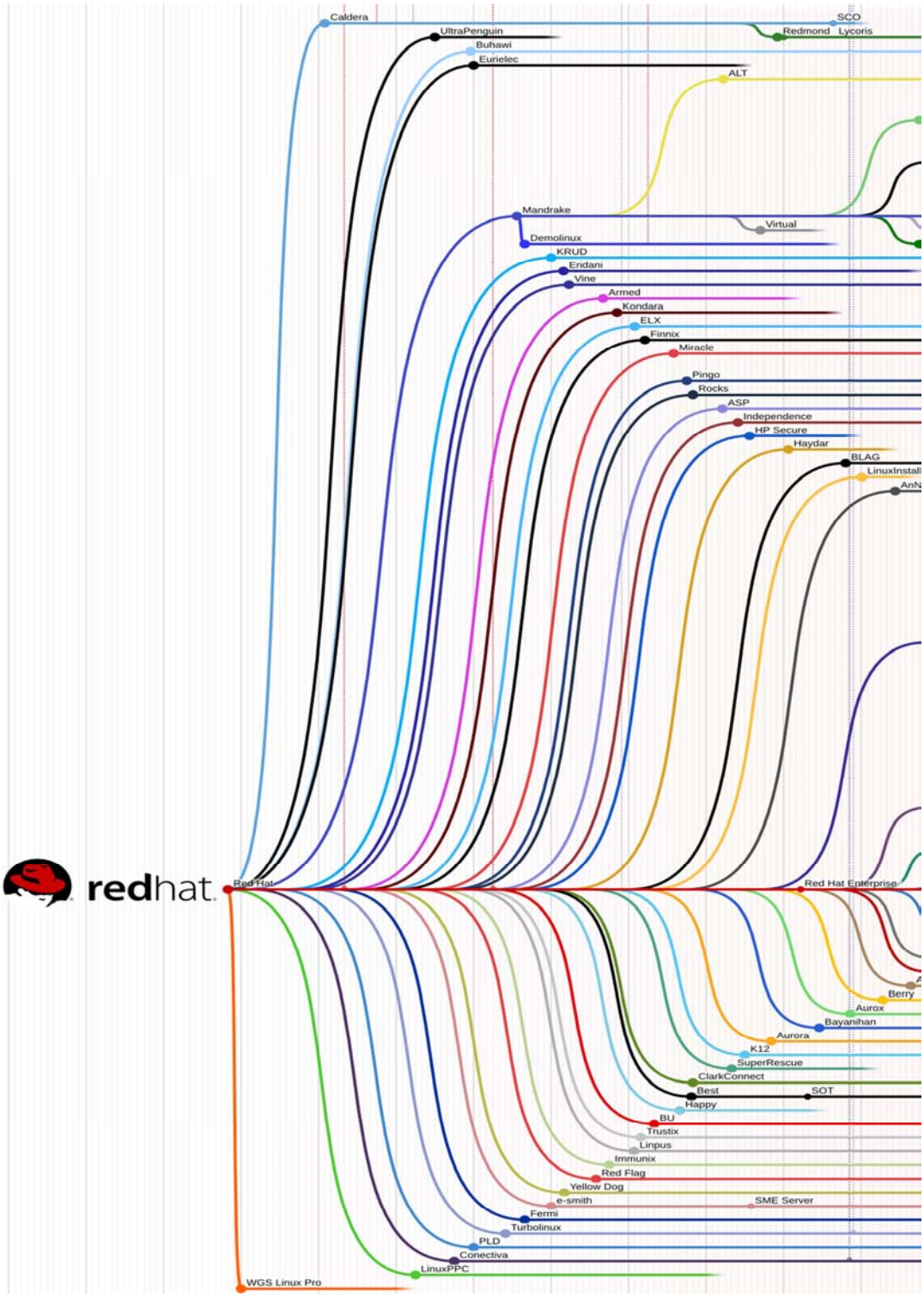


Σχήμα 3.4.2 Διανομές Slackware και SuSE (πηγή: futurist.se/gldt/)

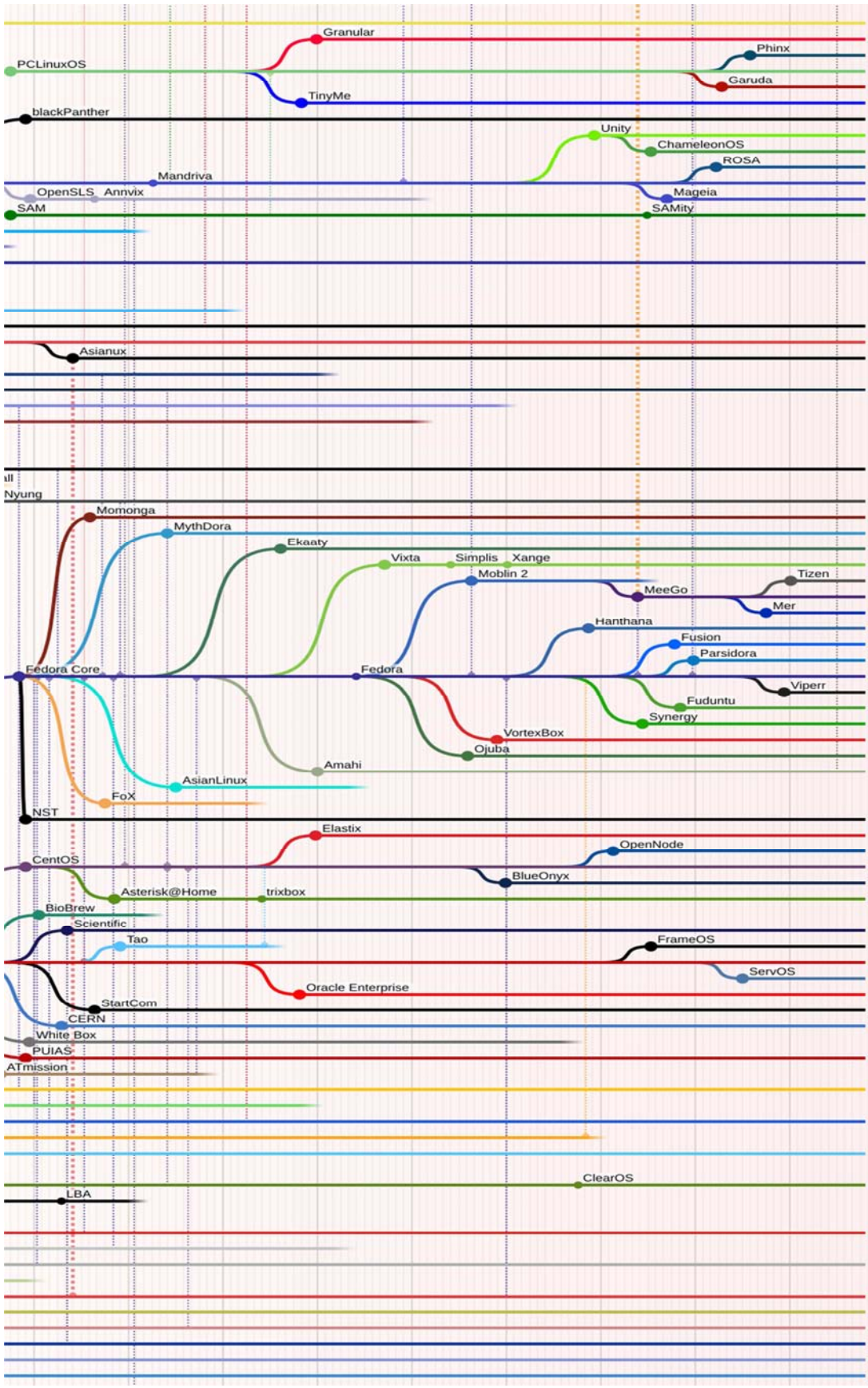
3.1.2.3 Η διανομή RedHat

Η διανομή του Redhat η οποία εισήγαγε τον διαχειριστή πακέτων τύπου .rpm ήταν μία ελεύθερη διανομή Linux που ξεκίνησε στις 3 Νοεμβρίου 1994. Το 2003 η εταιρία παραγωγής της διανομής αποφάσισε να μην υποστηρίξει περεταίρω την διανομή του Redhat και να ασχοληθεί με την εμπορική διανομή του. Μία άλλη έκδοση που βασίζονταν στο Red hat έκανε την εμφάνισή της στις 5/11/2003, η γνωστή διανομή Fedora η οποία χρηματοδοτείτε από την εταιρία Redhat (το έργο

ονομάζεται fedora project). Πολλές είναι και οι διανομές που στηρίζονται τόσο στο Redhat όσο και στο Fedora. Βασικές διανομές που στηρίζονται στο Redhat είναι οι: Oracle enterprise, το Cent OS, το PCLinux OS κλπ. Ενώ στο Fedora βασίζονται οι διανομές: MeeGo, Tizen κλπ. Στα σχήματα 3.5.1 και 3.5.2 παρουσιάζονται μερικές από τις διανομές που προέρχονται από την διανομή Redhat και Fedora.



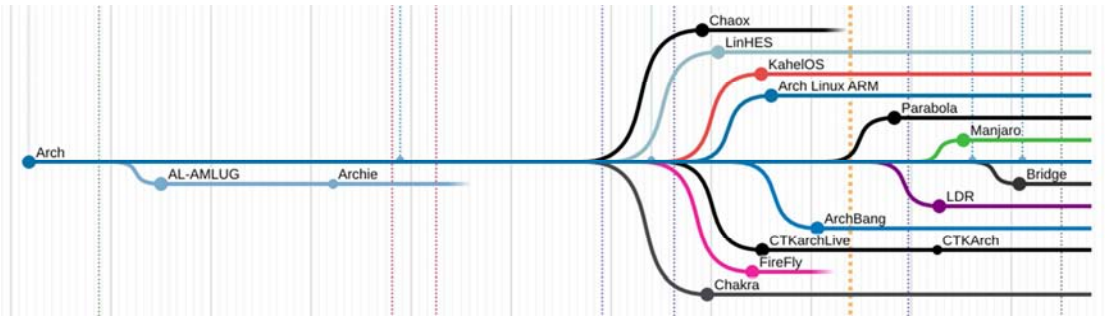
Σχήμα 3.5.1 Διανομές Red hat (πηγή: futurist.se/gldt/)



Σχήμα 3.5.2 Διανομές Red hat - Fedora (πηγή: futurist.se/gldt/)

3.1.2.4 Η διανομή Arch

Μία διανομή που ξεκίνησε αργά αλλά αποτελεί μία αρκετά ενδιαφέρουσα επιλογή είναι το Arch. Ξεκίνησε στις 11 Μαρτίου 2002 και είναι ανεξάρτητο από τις υπόλοιπες διανομές, τουτέστιν δεν βασίζεται σε καμία από τις παλαιότερες διανομές. Κύριά του χαρακτηριστικά είναι η παραμετροποίηση που ο χρήστης μπορεί να κάνει εξ’ αρχής/κατά την εγκατάσταση (η οποία χαρακτηρίζεται από μεγάλο βαθμό δυσκολίας για εκείνους που δεν είναι εξοικειωμένοι με το command line – ο χρήστης μπορεί να κάνει επιλογές για το τι θέλει και τι δεν θέλει να εγκαταστήσει) καθώς επίσης και το γεγονός ότι δεν διαθέτη διαφορετικές εκδόσεις κατά καιρούς στους χρήστες της αλλά μέσω του update του συστήματος γίνεται η ανανέωση όλων των στοιχείων και προγραμμάτων που έχουν εγκατασταθεί. Το μοντέλο ενημερώσεων του Arch Linux είναι rolling-release, οπότε μετά την ενημέρωση του συστήματος όλες οι εφαρμογές βρίσκονται στην τελευταία διαθέσιμη έκδοση. Αυτό μπορεί να ακούγεται ενδιαφέρον διότι έχοντας την τελευταία έκδοση των πακέτων είμαστε και περισσότερο καλυμμένοι από άποψη ασφαλείας αλλά τα νέα πακέτα ίσως να επιδράσουν στην σταθερότητα του συστήματός μας, μιας και οι νεότερες έκδοσης δεν έχουν δοκιμαστεί αρκετά. Κάτι τέτοιο έχει αναφερθεί από πολλούς χρήστες που χρησιμοποιούν την συγκεκριμένη διανομή και γι’ αυτό το Arch ίσως να μην είναι και το πλέον κατάλληλο για την χρησιμοποίηση του ως Server. Το Arch Linux έχει δεχθεί επίσης επιθέσεις για την έλλειψη υπογεγραμμένων πακέτων. Τα πακέτα και τα μεταδεδομένα τους δεν ελέγχονται για την αυθεντικότητά τους κατά την εγκατάσταση, με αποτέλεσμα να είναι δυνατόν να υπάρχουν κακόβουλοι mirror που μπορούν να διακινδυνεύσουν την ασφάλεια ενός συστήματος. Τέλος θα πρέπει να αναφερθεί πως για την εγκατάσταση, απεγκατάσταση, και ενημέρωση των προγραμμάτων στο Arch Linux γράφτηκε ένας ειδικός διαχειριστής πακέτων που ονομάζεται pacman. Το σχήμα 3.6 παρουσιάζει τις διανομές που προέρχονται από το Arch



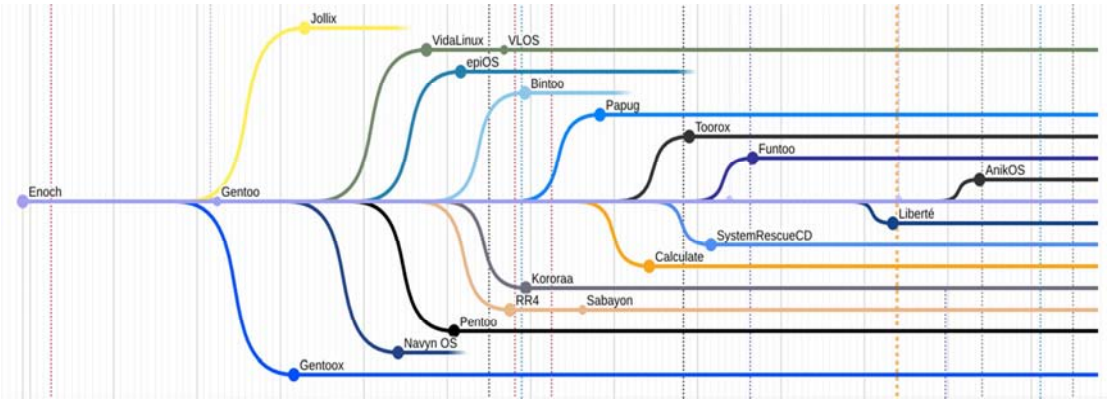
Σχήμα 3.6 Διανομές Arch (πηγή: futurist.se/gldt/)

3.1.2.5 Η διανομή Enoch/Gentoo

Το Gentoo πρώην Enoch είναι μία διανομή που κατασκευάστηκε με τη χρήση του συστήματος διαχείρισης πακέτων Portage. Σε αντίθεση με μια δυαδική διανομή λογισμικού, ο πηγαίος κώδικας συντάσσεται (compiled) σε τοπικό επίπεδο ανάλογα με τις προτιμήσεις του χρήστη και συχνά βελτιστοποιείται για το συγκεκριμένο τύπο υπολογιστή. Μεταγλωττισμένα εκτελέσιμα πακέτα είναι διαθέσιμα για κάποια

δημοφιλή προγράμματα ή για κάποια με ακυκλοφόρητο πηγαίο κώδικα. Το όνομα, προέρχεται από το gentoo penguin δηλαδή το γρήγορο κολύμπι και επελέγη ώστε να αντικατοπτρίζει τις πιθανές βελτιώσεις ταχύτητας στο υλικό διαμέσου του λογισμικού. Ο διαχειριστής πακέτων του Gentoo έχει σχεδιαστεί έτσι ώστε να είναι modular, φορητός, εύκολος στην διατήρηση και ευέλικτος. Το Gentoo μερικές φορές περιγράφεται ως μετα-διανομή, λόγω της σχεδόν απεριόριστης προσαρμοστικότητάς του, και εκ του γεγονότος ότι η πλειονότητα των χρηστών έχουν διαμορφώσει και έχουν εγκατεστημένα προγράμματα τα οποία είναι μοναδικά για εκείνους.

Πολλές άλλες μικρότερες και μεγαλύτερες διανομές υπάρχουν, όμως η ανάλυση τους και ίσως μόνο η αναφορά τους δεν θα μπορούσε να είναι εφικτή καθώς υπάρχουν πάνω από 270 διαφορετικές διανομές και θα ήταν αδύνατον να αναλυθούν και να παρουσιαστούν όλες. Στο σχήμα 3.7 μπορούμε να δούμε τις διανομές που προέρχονται από το Enoch/Gentoo.



Σχήμα 3.7 Διανομές Enoch/Gentoo (πηγή: futurist.se/gldt/)

3.2 Σχεδιασμός και υλοποίηση του Linux και του Kernel

Σε αντίθεση με τις περισσότερες σύγχρονες έρευνες για τα λειτουργικά συστήματα, η προσέγγιση του Linux σχετικά με τα λειτουργικά συστήματα είναι πολύ ρεαλιστική, και αντί να επικεντρώνονται στο σχεδιασμό ενός νέου πυρήνα το επίκεντρο του ήταν και είναι η σταθερή και αποτελεσματική του εφαρμογή (**Torvalds**, 1997). Ο σχεδιασμός του πυρήνα του Linux βασίζεται σε τρία βασικά ζητήματα σχεδιασμού που έχουν άμεσα επηρεάσει την εφαρμογή του. Τα κύρια θέματα πάνω στα οποία στηρίζεται είναι:

- ✓ Απλότητα. Το προφανές αυτό πράγμα μερικές φορές αγνοείται, αλλά ένας πυρήνας λειτουργικού συστήματος είναι μια σύνθετη οντότητα που πρέπει να είναι σε θέση να λειτουργεί κάτω από ανεξέλεγκτες συνθήκες και ενδεχομένως ακόμα και ένα εχθρικό περιβάλλον. Τα προγραμματιστικά σφάλματα στα λειτουργικά συστήματα είναι λιγότερο αποδεκτά από ότι στα προγράμματα/εφαρμογές, καθώς επίσης και τα θέματα ασφάλειας είναι υψίστης σημασίας. Ένας σύνθετος σχεδιασμός είναι πιο δύσκολος να επαληθευθεί είτε για σφάλματα ή θέματα ασφάλειας, οπότε η απλότητα των βασικών υπηρεσιών είναι απαραίτητη.

- ✓ Αποδοτικότητα. Ο πυρήνας ασχολείται με σχεδόν όλες τις δραστηριότητες του υλικού, και ως εκ τούτου πρέπει να είναι αρκετά αποτελεσματικός έτσι ώστε ποτέ να μην περιορίζεται η απόδοσή του.
- ✓ Συμβατότητα. Ενώ οι βασικές λειτουργίες ενός πυρήνα είναι αρκετά ενδιαφέρουσες για τους ερευνητές, οι περισσότεροι άνθρωποι δεν θέλουν να γνωρίζουν τι συμβαίνει στο εσωτερικό του, εφόσον τα προγράμματά τους λειτουργούν αποτελεσματικά. Ως εκ τούτου, ένα από τα πιο σημαντικά χαρακτηριστικά ενός λειτουργικού συστήματος είναι η προσφορά στον χρήστη ευκολιών έτσι ώστε ο ίδιος να μην ασχολείται με πράγματα τα οποία δεν γνωρίζει ή θα έπρεπε να είναι προγραμματιστής για να τα αντιμετωπίσει.

Τα θέματα αυτά του σχεδιασμού, σε συνδυασμό με μια ρεαλιστική προσέγγιση του προγραμματισμού, έχουν οδηγήσει σε ένα σύστημα που μοιράζεται τα χαρακτηριστικά τόσο των παραδοσιακών μονολιθικών πυρήνων αλλά και των νεότερων microkernels (η ανάλυσή τους έγινε στο 1^{ο} κεφάλαιο). Η υλοποίηση και ο σχεδιασμός του microkernel απορρίφθηκε στο Linux λόγω αμφιβολιών για την αποτελεσματικότητα και την απλότητα, αλλά κατά τη διάρκεια της ανάπτυξης του μονολιθικού πυρήνα του Linux πολλά από τα χαρακτηριστικά που συνήθως συνδέονται με τους mikrokernels υλοποιήθηκαν.

Ο σχεδιασμός έχει επίσης επηρεαστεί έντονα από τη γενική διαθεσιμότητα του πηγαίου κώδικα του Linux στο Internet και ειδικά στις κοινότητες που ασχολούνται και συμβάλουν στην υλοποίησή του. Αυτό δεν έχει μόνο οδηγήσει σε μια πολύ δυναμική ανάπτυξη της κοινότητας που φυσικά έχει εξαπλωθεί σε όλο τον κόσμο, αλλά επίσης έχει άμεσα επηρεάσει τον ίδιο το σχεδιασμό του πυρήνα.

Αντί να επικεντρωθεί σε μια έννοια ενός δυαδικού πυρήνα που διασυνδέεται με τα υπόλοιπο στοιχεία (είτε υλικό, είτε λογισμικό), η προσέγγιση του Linux ήταν και είναι να υπάρχουν διαθέσιμες πηγές που μπορούν να χρησιμοποιηθούν για να δημιουργήσουν το δυαδικό κώδικα συστήματος που θέλουμε. Αυτή η προσέγγιση επιτρέπει την πραγματική αλληλεπίδραση με το σύστημα, που πρέπει να προσδιορίζονται κατά τη μεταγλώττιση και όχι κατά το χρόνο εκτέλεσης.

Με τη χρήση του πραγματικού χρόνου εκτέλεσης δυναμικών συνδέσεων στις οντότητες του πυρήνα, ο χρήστης μπορεί να προσθέσει και να αφαιρέσει χαρακτηριστικά, όμως η διαθεσιμότητα και το compiled των πηγών αυξάνει πάντα τη την απόδοση του συστήματος καθορίζοντας τις ρυθμίσεις κατά τη μεταγλώττιση και όχι κατά τον χρόνο εκτέλεσης.

3.2.1 Εφαρμογή

Ενώ τα περισσότερα σύγχρονα λειτουργικά συστήματα τείνουν να ευνοούν μια microkernel σχεδίαση και αντικειμενοστραφή (**Torvalds**, 1997) προσανατολισμένη, το Linux έχει περιέλθει προς την κατεύθυνση ενός πολύ πιο παραδοσιακού σχεδιασμού. Όπως και με το UNIX που προσπαθεί να είναι συμβατό, το Linux χρησιμοποιεί ένα παραδοσιακό μονολιθικό πυρήνα κυρίως γραμμένο στην φορητή γλώσσα ANSI C.

Η C είναι γενικά κατάλληλη για τον προγραμματισμό των συστημάτων (άλλωστε αυτή ήταν και μία από τις αρχικές της χρήσεις) ο πυρήνας του Linux

επιπλέον χρησιμοποιεί ορισμένα χαρακτηριστικά που προσφέρονται από το μεταγλωττιστή GNU C που τον καθιστούν ευκολότερο στην εφαρμογή ορισμένων χαρακτηριστικών αποτελεσματικά. Οι δύο σημαντικές επεκτάσεις που χρησιμοποιούνται στη C είναι οι ενσωματωμένες λειτουργίες και η συμβολική γλώσσα που εντάσσεται στη C.

Εκτός από τον κώδικα που είναι γραμμένος στη C, ορισμένες ρουτίνες χαμηλού επιπέδου έχουν γραφτεί απευθείας σε συμβολική γλώσσα, είτε για λόγους απόδοσης ή για λόγους συνθηκών όπου η σημασιολογία της C δεν είναι σε θέση να καλύψει τις ακριβείς απαιτήσεις παραγωγής κώδικα (συνήθως οι απαιτήσεις επιβάλλονται από το ίδιο το υλικό, όπως τα σφάλματα χειρισμού των ρουτινών). Ο παραδοσιακός σχεδιασμός επιλέχθηκε λόγω των σοβαρών αμφιβολιών σχετικά με την απόδοση και την απλότητα των λειτουργικών συστημάτων που βασίζονται στους microkernels.

Ενώ η απλότητα έχει χρησιμοποιηθεί ως ένα από τα επιχειρήματα για τους microkernels, το επιχείρημα αυτό φαίνεται αμφίβολο καθώς υπάρχουν πολύπλοκες αλληλεπιδράσεις μεταξύ των διαφόρων modules. Αυτό το ζήτημα γενικά επιδεινώνεται από τα πολύπλοκα ζητήματα απόδοσης, και η βασική απλότητα του microkernel είναι συχνά κυριευμένη από την πολυπλοκότητα των θεμάτων αυτών.

Ενώ αυτά τα επιχειρήματα του Linus προσπαθούν να επιχειρηματολογήσουν εναντίον των microkernel, ο Linus θεωρεί επίσης ότι τα πλεονεκτήματα των microkernels έχουν υπερεκτιμηθεί στη βιβλιογραφία, και ότι μια πιο απλή παραδοσιακή προσέγγιση μπορεί να εξυπηρετήσει τις ανάγκες των χρηστών καλύτερα. Οι microkernels τείνουν να είναι καταλληλότερο για κατανεμημένα συστήματα παρά για λειτουργικά τύπου UNIX.

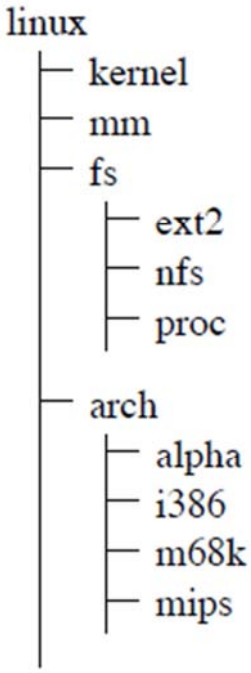
Ακόμη και ένας μονολιθικός πυρήνας UNIX μπορεί να θεωρηθεί ως ένας μικρός microkernel καθώς όλες οι υπηρεσίες διεξάγονται από τους δαίμονες του συστήματος του UNIX. Ενώ πολλά άλλα λειτουργικά συστήματα τείνουν να διαχειρίζονται την επαλήθευση/έλεγχο αυθεντικότητας χρήστη και τον χειρισμό των συνεδριών στον πυρήνα (και σε ορισμένες περιπτώσεις ακόμη και στα γραφικά), η νοοτροπία αυτή είναι ενάντια στην ίδια τη φιλοσοφία του UNIX, όπου οι υπηρεσίες αυτές παρέχονται μέσω δαιμόνων στο επίπεδο χρήστη.

3.2.2 Οργάνωση του πυρήνα

Σύμφωνα με τον δημιουργό του πυρήνα του Linux (**Torvalds**, 1997), ο βασικός πυρήνας του Linux είναι άμεσα οργανωμένος γύρω από τις πρωτογενείς υπηρεσίες που παρέχει: χειρισμό των διαδικασιών, διαχείριση μνήμης, διαχείριση του συστήματος αρχείων, πρόσβαση στο δίκτυο και τα προγράμματα οδήγησης για το υλικό (έχουν αναλυθεί και στο κεφάλαιο 2, αφού το Linux και το Unix παρουσιάζουν αρκετές ομοιότητες). Οι τομείς αυτοί αντιστοιχούν στους πηγαίους καταλόγους του πυρήνα, mm, fs, net και στους οδηγούς αντίστοιχα. Επιπλέον, αυτές οι περιοχές συχνά υποδιαιρούνται σε συγκεκριμένες υπηρεσίες (σχήμα 3.8).

Το βασικό αξίωμα της φορητότητας του Linux είναι η ενιαία μοναδική πηγή (single source concept), δηλαδή όλες οι αρχιτεκτονικές μοιράζονται τον ίδιο βασικό δενδρικό πηγαίο κώδικα. Η εξαρτήσεις του κώδικα της αρχιτεκτονικής είναι διαθέσιμες στον κατάλογο arch ο οποίος έχει υποκαταλόγους για κάθε αρχιτεκτονική

που υποστηρίζεται. Αν και δεν υπάρχουν απόλυτοι κανόνες για το τι ακριβώς μια αρχιτεκτονική έχει να προσφέρει, κοινά πρότυπα έχουν γενικά ως αποτέλεσμα την παρόμοια οργάνωση της αρχιτεκτονικής των υποκαταλόγων, και οι περισσότερες από αυτές ακολουθούν μια εγκατάσταση με ξεχωριστούς καταλόγους για τη διαδικασία χειρισμού του πυρήνα, τη διαχείριση μνήμης (mm) και τις ρουτίνες της βιβλιοθήκης που έχουν βελτιστοποιηθεί για τη συγκεκριμένη αρχιτεκτονική (lib).



Σχήμα 3.8 Οργάνωση του πυρήνα

Αυτή η ιεραρχική υποδιαίρεση των πηγών όχι μόνο καθιστά ευκολότερη την επισκόπηση του συστήματος, αλλά λειτουργεί επίσης και για την διαμόρφωση/συντήρηση/προσαρμογή του πυρήνα. Με την βοήθεια αυτής της μεθόδου πολλοί προγραμματιστές είναι δυνατό να δουλεύουν σε διαφορετικές περιοχές του πυρήνα χωρίς να χρειάζεται να ανησυχούν για το τι γίνεται στις υπόλοιπες περιοχές.

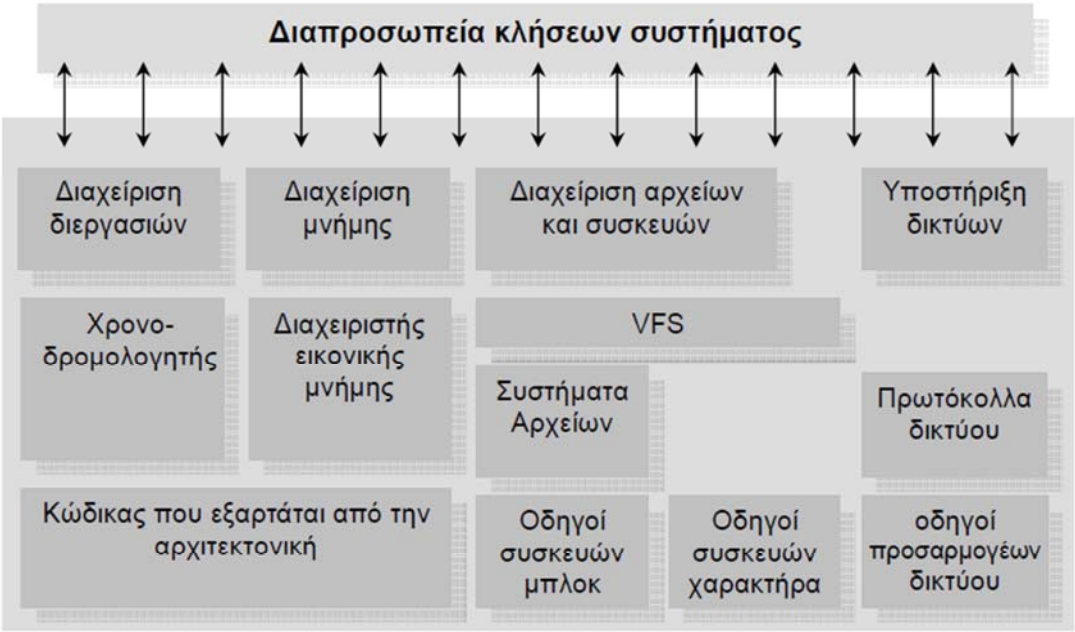
Η ιεραρχική/σπονδυλωτή (modular) υποδιαίρεση του πυρήνα έχει επίσης οδηγήσει σε περιορισμό του αντικειμενοστρεφούς σχεδιασμού. Ενώ η γλώσσα C που χρησιμοποιείται από τον πυρήνα, εγγενώς δεν υποστηρίζει αυτό το είδος του προσανατολισμού προς τα αντικείμενα, τα αποτελέσματα μπορεί να είναι παρόμοια με τη χρήση της λειτουργίας δεικτών και των κοινών δομών δεδομένων. Αυτό είναι ιδιαίτερα παρατηρήσιμο στον κώδικα του συστήματος αρχείων, όπου το εικονικό στρώμα συστήματος αρχείων που υλοποιεί το γενικό σημασιολογικό σύστημα αρχείων του UNIX μπορεί και αναγνωρίζει τις συγκεκριμένες χαμηλού επιπέδου υλοποιήσεις.

3.2.3 Linux Kernel Modules

Ο πυρήνας του Linux έχει τη δυνατότητα να φορτώνει και να αφαιρεί από το χώρο διευθύνσεων του πυρήνα τμήματα κώδικα, όποτε αυτό είναι αναγκαίο (**Κούκης**, 2002). Τα τμήματα κώδικα αυτά ονομάζονται αρθρώματα του πυρήνα του Linux, ή kernel modules. Ο κώδικας που περιέχεται σε κάποιο module του πυρήνα εισάγεται στον εικονικό χώρο διευθύνσεων ενός πυρήνα που βρίσκεται ήδη σε λειτουργία και από τη στιγμή που θα συμβεί αυτό δεν υπάρχει καμία διάκριση ανάμεσα σε αυτόν και τον κώδικα του πυρήνα που προϋπήρχε. Έτσι, εκτελείται επίσης στην προνομιούχο κατάσταση της κεντρικής μονάδας επεξεργασίας και έχει πλήρη πρόσβαση σε κάθε εσωτερική δομή δεδομένων του πυρήνα και στο χώρο διευθύνσεων κάθε διεργασίας, όπως θα περίμενε κανείς από ένα μονολιθικό σύστημα. Παρόλο που δεν υπάρχει κανένας περιορισμός ως προς τι θα μπορούσε να υλοποιηθεί ως module, επιλέγεται να υλοποιούνται ως modules οδηγοί για συσκευές υλικού αλλά και εικονικές συσκευές, συστήματα αρχείων και πρωτόκολλα δικτύων. Η ύπαρξη του μηχανισμού των modules στον πυρήνα του Linux παρέχει ορισμένα σημαντικά πλεονεκτήματα. Διευκολύνεται σε μεγάλο βαθμό η ανάπτυξη νέων επεκτάσεων για τον πυρήνα, εφόσον αυτές μπορούν να δοκιμάζονται σε έναν πυρήνα που ήδη εκτελείται, χωρίς να απαιτείται ένας χρονοβόρος κύκλος αλλαγής του κώδικα – εκ νέου μεταγλώττιση ολόκληρου του πυρήνα - επανεκκίνηση του συστήματος με τον νέο πυρήνα.

Ο νέος κώδικας μεταγλωττίζεται χωριστά, εισάγεται ως module, ελέγχεται και αν χρειάζονται επιπλέον αλλαγές αφαιρείται και μεταβάλλεται χωρίς να επηρεαστεί η λειτουργία του υπόλοιπου συστήματος. Επιπλέον, εξοικονομείται κεντρική μνήμη και μειώνεται το μέγεθος του πυρήνα, εφόσον η λειτουργία του συστήματος ξεκινά με τους απολύτως απαραίτητους οδηγούς συσκευών ενσωματωμένους στον πυρήνα, ενώ οι οδηγοί για συσκευές που χρησιμοποιούνται σπάνια μπορούν να φορτώνονται όποτε υπάρξει ανάγκη χρήσης τους. Οι οδηγοί αυτοί παραμένουν στη μνήμη προσωρινά, φορτωμένοι ως modules και αφαιρούνται όταν δεν χρησιμοποιούνται. Γίνεται λοιπόν φανερό ότι ο μηχανισμός των modules προσφέρει στο Linux πλεονεκτήματα τα οποία χαρακτηρίζουν κυρίως τα συστήματα που ακολουθούν τη σχεδιαστική επιλογή του μικροπυρήνα, όπως είναι η διάκριση του κώδικα σε ξεχωριστά τμήματα και η δομημένη οργάνωση του, διατηρώντας όμως το κύριο πλεονέκτημα μιας μονολιθικής σχεδίασης, που είναι ο ενιαίος χώρος διευθύνσεων πυρήνα. Αυτό γίνεται δυνατό λόγω του τρόπου με τον οποίο φορτώνεται ο κώδικας ενός kernel module στη μνήμη Κατά τη φόρτωση ενός module, που είναι στην ουσία αντικειμενικός κώδικας με αναφορές σε άλλα τμήματα του πυρήνα, γίνεται δυναμική σύνδεσή του με τον υπόλοιπο κώδικα. Χρησιμοποιώντας έναν πίνακα συμβόλων, ο οποίος περιέχει τις διευθύνσεις όλων των βασικών συναρτήσεων και δομών δεδομένων που είναι δυνατόν να χρησιμοποιηθούν από κάποιο module, επιλύονται οι αναφορές που γίνονται από το module σε σημεία έξω από αυτό. Η διαδικασία αυτή γίνεται μόνο μία φορά, κατά την εισαγωγή του στο χώρο πυρήνα. Στη συνέχεια, ο κώδικας του module εκτελεί απλές κλήσεις συναρτήσεων και μεταβάλλει απευθείας τις δομές του πυρήνα, χωρίς την ανάγκη για χρονοβόρα ανταλλαγή μηνυμάτων. Επιπλέον, υπάρχει η δυνατότητα το ίδιο το module να εξάγει κάποια από τα σύμβολά του ώστε ένα επόμενο να μπορεί να συνδεθεί με αυτά. Έτσι, υλοποιείται μια στοίβα, και κάθε ένα από τα αρθρώματα που περιέχονται σε αυτή χρησιμοποιεί υπηρεσίες αυτού που βρίσκεται σε χαμηλότερο επίπεδο.

Συνοπτικά, η δομή του πυρήνα του Linux και η διάκριση των τμημάτων του ανάλογα με τη λειτουργία που επιτελούν παρουσιάζεται στο παρακάτω διάγραμμα:



Σχήμα 3.9 Δομή του πυρήνα του Linux

Ο πυρήνας του Linux δεν θα αναλυθεί περισσότερο διότι έχει την ίδια περίπου δομή και κάνει τις ίδιες εργασίες με εκείνων του Unix ο οποίος αναλύθηκε στο κεφάλαιο 2. Επίσης υπάρχει εκτενείς βιβλιογραφία και εργασίες οι οποίες υπερκαλύπτουν το συγκεκριμένο κομμάτι, αντιθέτως για το υπόλοιπο τις διπλωματικής εργασίας θα επικεντρωθούμε σε θέματα ασφαλείας και ευπαθειών που παρουσιάζει ιδιαίτερο ενδιαφέρον καθώς και στην εγκατάσταση και παραμετροποίηση της διανομής Slackware.

3.3 Ευπάθειες στον πυρήνα του Linux

Ο πυρήνας είναι ένα μέρος του συστήματος το οποίο είναι έμπιστο στα περισσότερα συστήματα (**Chen et al**. όπως αναφέρεται στους **Arnold et al**, **Mokhov et al**, **Christiey** και **Martin**). Θέματα ευπαθειών και ασφάλειας μπορεί να επιτρέψουν σε ένα άτομο να παρακάμψει τυχόν μηχανισμούς προστασίας του πυρήνα και να αποκτήσει διαχειριστικά δικαιώματα πρόσβασης (root privileges και root access) το οποίο είναι μεγίστης σημασίας στο Linux καθώς ο διαχειριστής του συστήματος έχει απεριόριστες δυνατότητες. Μέσα από αυτή την ενότητα θα προσπαθήσουμε να ερευνήσουμε και να αναλύσουμε τα τρωτά σημεία του πυρήνα του Linux στο παρελθόν, την κατηγοριοποίηση τους, την αξιολόγηση, τις αμυντικές τεχνικές που θα μπορούσαν να είχαν χρησιμοποιηθεί για την πρόληψή τους, και για το ποιά είναι τα εκκρεμούντα προβλήματα και ευπάθειες.

Αρκετές μελέτες έχουν διεξαχθεί προκειμένου να κατανοήσουν τους τύπους των τρωτών σημείων του πυρήνα που εμφανίζονται στην πράξη. Μια μελέτη από τους Arnold et al υποστηρίζει ότι κάθε σφάλμα του πυρήνα θα πρέπει να αντιμετωπίζονται ως ένα κρίσιμο κενό ασφαλείας και θα πρέπει να επιδιορθώνετε το συντομότερο δυνατό. Οι Mokhov et al. διερεύνησαν πώς οι προγραμματιστές του πυρήνα επουλώνουν με paches τα γνωστά τρωτά σημεία. Οι Christiey και Martin παραθέτουν μία έκθεση σχετικά με τις διανομές που έχουν τις περισσότερες ευπάθεια στο γνωστό site ευπαθειών CVE. Καμία από τις μελέτες δεν ρίχνει φως στο τι τεχνικές θα μπορούσαν να χρησιμοποιηθούν για την πρόληψη άγνωστων ευπαθειών. Παρακάτω παρατίθενται μελέτες περίπτωσης των τρωτών σημείων του πυρήνα του Linux που ανακαλύφθηκαν από τον Ιανουάριο 2010 έως τον Μάρτιο του 2011.

Αυτά τα τρωτά σημεία τα κατηγοριοποιούμε κατά είδος, σχετικά με τα λάθη που κάνουν οι προγραμματιστές και τον αντίκτυπο που έχουν στην ασφάλεια των συστημάτων. Με βάση αυτή τη λίστα των τρωτών σημείων του πυρήνα, μπορούμε να διεξάγουμε μια δεύτερη μελέτη περίπτωσης, εξετάζοντας το κατά πόσο αποτελεσματικές είναι οι τεχνικές που προτείνονται από τους ερευνητές και θα μπορούσαν να μετριάσουν τα τρωτά σημεία του πυρήνα του Linux. Αυτές οι τεχνικές περιλαμβάνουν μηχανισμούς εκτέλεσης ελέγχων, όπως τον έλεγχο τις ακεραιότητας του κώδικα, την απομόνωση του λογισμικού με τα σφάλματα, τους οδηγούς οδήγησης των συσκευών σε επίπεδο χρήστη καθώς και τα εργαλεία εύρεσης των διαφόρων bugs και την στατική ανάλυση. Η εξέταση αυτή δεν είναι εμπειρική, αλλά βασίζεται αποκλειστικά στην κατανόηση των τεχνικών.

Τα ευρήματα των δύο μελετών μας έχουν ως εξής. Πρώτον, βρίσκουμε ότι υπάρχουν 10 κοινές κατηγορίες τρωτών σημείων του πυρήνα στο Linux, η οποία μπορεί να οδηγήσει σε επιθέσεις που κυμαίνονται από αυθαίρετες τροποποιήσεις μνήμης έως διαρροές πληροφοριών και επιθέσεις denial-of-service (DDoS attacks). Δεύτερον, διαπιστώνουμε ότι περίπου 2 = 3 από τα τρωτά σημεία βρίσκονται στα modules του πυρήνα ή τους οδηγούς και ότι αντίθετα, 1 = 3 από τα σφάλματα βρίσκονται στο κέντρο του πυρήνα. Τρίτον, διαπιστώνουμε ότι καμία υπάρχουσα τεχνική δεν μπορεί να κάνει πρόληψη όλων των τρωτών σημείων του πυρήνα, και ότι μια τεχνική συχνά εκμεταλλεύεται από μερικούς για δημιουργία άλλων ευπαθειών. Τέταρτον, υπάρχουν ορισμένες κατηγορίες ευπαθειών που δεν εξετάζονται καθόλου. Για παράδειγμα, σημασιολογικά τρωτά σημεία, όπου παραβιάζονται σταθερές ασφαλείας υψηλού επιπέδου, είναι δύσκολο να πιαστούν με state-of-the-art τεχνικές

που εστιάζουν ως επί το πλείστον στην ασφάλεια της μνήμης και την ακεραιότητα του κώδικα.

Ο πίνακας 3.1 κατηγοριοποιεί τις 141 ευπάθειες του πυρήνα του Linux που δημοσιεύονται στον κατάλογο CVE από τον Ιανουάριο 2010 έως τον Μάρτιος 2011 και το είδος των επιθέσεων που μπορεί να εκμεταλλευτούν οι ευπάθειες αυτές. Παρά την ποικιλομορφία τους, τα περισσότερα από αυτά τα τρωτά σημεία χωρίζονται σε 10 κατηγορίες, με βάση το είδος των λαθών προγραμματισμού που οι προγραμματιστές έκαναν, όπως αναφέρονται στην πρώτη στήλη. Δεδομένου ότι κάθε ευπάθεια μπορεί συχνά να αξιοποιηθεί με διάφορους τρόπους, κατηγοριοποιούμε επιπλέον τους διαφόρους τρόπους εκμετάλλευσης σε διάφορες κατηγορίες επιθέσεων. Η παραφθορά της μνήμης επιτρέπει συνήθως σε έναν αντίπαλο να εκτελέσει αυθαίρετες ενέργειες στον πυρήνα, και έτσι υπάρχει ένα υπερσύνολο από άλλους τύπους επιθέσεων, όπως η παραβίαση των πολιτικών, DDoS, αποκάλυψη πληροφοριών κλπ. Ως εκ τούτου, εάν μια ευπάθεια οδηγεί σε μια παραφθορά στη μνήμη, δεν το μετράμε στο πλαίσιο άλλων κατηγοριών. Στο υπόλοιπο αυτής της ενότητας αναλύουμε τις κατηγορίες των τρωτών σημείων που έχουμε βρει.

Ελλιπείς έλεγχος δεικτών (Missing pointer checks). Ο πυρήνας παραλείπει ελέγχους access_ok ή ενέργειες κατάχρησης "faster", όπως __get_user, η οποία δεν επικαιροποιεί τις τιμές των δεικτών που παρέχονται από το χρήστη ή τους δείκτες των μεταβλητών για να εξασφαλίσει ότι δείχνουν μόνο σε περιοχή της μνήμης χρήστη. Αυτά τα σφάλματα/κενά ασφαλείας επιτρέπουν σε μη προνομιούχες διαδικασίες να διαβάσουν ή να εγγράψουν αυθαίρετα σε θέσεις μνήμης του πυρήνα, οδηγώντας σε παραφθορά της μνήμης (CVE-2010-4258),

Ευπάθεια	Παραφθορές μνήμης	Παραβιάσεις πολιτικής	DDoS επιθέσεις	Αποκαλύψεις πληροφοριών	Διάφορα
Missing pointer check	6	0	1	2	0
Missing permission check	0	15	3	0	1
Buffer overflow	13	1	1	2	0
Integer overflow	12	0	5	3	0
Uninitialized data	0	0	1	28	0
Null dereference	0	0	20	0	0
Divide by zero	0	0	4	0	0
Infinite loop	0	0	3	0	0
Data race / deadlock	1	0	7	0	0
Memory mismanagement	0	0	10	0	0
Miscellaneous	0	0	5	2	1
Σύνολα	32	16	60	37	2

Πίνακας 3.1 Ευπάθειες (γραμμές) vs πιθανά exploits (στήλες) (πηγή: Chen et al)

Ευπάθεια	Σύνολα	Πυρήνας	Οδηγοί συσκευών	net	fs	sound
Missing pointer check	8	4	3	1	0	0
Missing permission check	17	3	1	2	11	0
Buffer overflow	15	3	1	5	4	2
Integer overflow	19	4	4	8	2	1
Uninitialized data	29	7	13	5	2	2
Null dereference	20	9	3	7	1	0
Divide by zero	4	2	0	0	1	1
Infinite loop	3	1	1	1	0	0
Data race / deadlock	8	5	1	1	1	0
Memory mismanagement	10	7	1	1	0	1
Miscellaneous	8	2	0	4	2	0
Σύνολα	141	47	28	35	24	7

Πίνακας 3.2 Ευπάθειες (γραμμές) vs περιοχές αποθήκευσης/κατάλογοι (στήλες) (πηγή: Chen et al)

Ελλιπείς έλεγχοι αδειών (Missing permission checks). Ο πυρήνας εκτελεί μια προνομιακή λειτουργία χωρίς να εξετάζει αν η καλούσα διεργασία έχει το προνόμιο να το πράξει. Ένα θέμα ευπάθειας/ασφαλείας σε αυτή την κατηγορία μπορεί να οδηγήσει σε μια παραβίαση της πολιτικής ασφαλείας του πυρήνας. Οι επιθέσεις που μπορούν να εκμεταλλευτούν αυτή την ευπάθεια εξαρτώνται από ποια είναι η πολιτική ασφάλειας και κυμαίνεται από τον αυθαίρετο κώδικα εκτέλεσης (CVE-2010 - 4347), την κλιμάκωση προνομίων (CVE-2010-2071 και CVE-2010-1146), και την αντικατάσταση ενός αρχείου προσάρτησης (CVE- 2010-2066 n CVE-2010-2537).

Υπερχείλιση (overflow) του buffer. Ο πυρήνας ελέγχει λανθασμένα το άνω ή κάτω όριο, κατά την πρόσβαση του στο buffer (CVE-2011-1010), κατανέμει ένα μικρότερο buffer από ότι υποτίθεται (CVE-2010-2492), χρησιμοποιεί ανασφαλείς λειτουργίες χειρισμού των μεταβλητών Strings (CVE-2010-1084), ή καθορίζει τις τοπικές μεταβλητές οι οποίες είναι πολύ μεγάλες για τις στοίβες του πυρήνα (CVE-2010-3848). Οι επιθέσεις που μπορεί να εκμεταλλευτούν αυτή την ευπάθεια είναι: η παραφθορές μνήμης (για εγγραφές), ή η αποκάλυψη πληροφοριών (για το διάβασμα). Κάποιοι κακόβουλοι χρήστες με προνόμια διαχειριστών μπορούν να εξαπολύσουν επιθέσεις, αντικαθιστώντας τους κοντινούς δείκτες λειτουργίας και υπονομεύοντας την ακεραιότητα της ροής ελέγχου του πυρήνα

Υπερχείλιση ακεραίων (Integer overflow). Ο πυρήνας εκτελεί μία λανθασμένη λειτουργία ακεραίων, με αποτέλεσμα την υπερχείλιση ακεραίων, υποχείληση, ή κατά την αναφορά των σφαλμάτων. Οι κακόβουλοι χρήστες μπορούν να ξεγελάσουν τον πυρήνα έτσι ώστε να χρησιμοποιήσει την εσφαλμένη τιμή προκειμένου να κατανείμει ή να έχει πρόσβαση στη μνήμη, επιτρέποντας σε παρόμοιες επιθέσεις που επιτρέπονται από την "υπερχείλιση buffer". Για παράδειγμα, η υπερχείλιση μετά από πολλαπλασιασμούς μπορεί να προκαλέσει την διάθεση από

τον πυρήνα ενός μικρότερου ποσού από αυτό που απαιτείτε στο Buffer (CVE-2010-3442), η υποχείληση μετά την αφαίρεση μπορεί να προκαλέσει παραφθορά μνήμης μετά από το τέλος του buffer (CVE-2010-3873) και τέλος κατά την διαδικασία αναφοράς σφαλμάτων μπορεί να παρακάμψει τα όρια ελέγχου και προκαλέσει αποκάλυψη πληροφοριών (CVE-2010 - 3437).

Προετοιμασία δεδομένων (Uninitialized data.). Τα περιεχόμενα του buffer του πυρήνα στο χώρο χρήστη δεν μηδενίζουν τους αχρησιμοποίητους τομείς και έτσι γίνεται διαρροή δυνητικά ευαίσθητων πληροφοριών προς τις διεργασίες του χρήστη, όπως οι μεταβλητές στη στοίβα πυρήνα (CVE-2010-3876). Αυτή η κατηγορία έχει 29 τρωτά σημεία, που την κατατάσσει στην υψηλότερη όλων των κατηγοριών. Μια άμεση επίθεση που χρησιμοποιεί αυτή την ευπάθεια οδηγεί σε ακούσια αποκάλυψη πληροφοριών. Ωστόσο, ευπάθειες στην κατηγορία αυτή μπορούν να επιτρέψουν άλλες επιθέσεις, όπως επιθέσεις που απαιτούν να γνωρίζουν την ακριβή διεύθυνση/διαδρομή της δομής δεδομένων του πυρήνα, τα ιδιωτικά κλειδιά πυρήνα, ή άλλα τυχαία γεγονότα του πυρήνα

Κακοδιαχείριση της μνήμης (Memory mismanagement). Αυτή η κατηγορία περιλαμβάνει τρωτά σημεία στη διαχείριση της μνήμης του πυρήνα, όπως εξωγενή κατανάλωση μνήμης (CVE-2011-0999), απώλεια μνήμης (CVE-2010-4249), Double free (CVE-2010-3080). Για τις ευπάθειες που εξετάσαμε, ένας κακόβουλος χρήστης μπορεί να εξαπολύσει επιθέσεις DDoS με την αξιοποίηση τους, αν και σε γενικές γραμμές είναι δυνατόν να επιτευχθεί αυθαίρετη καταστροφή της μνήμης.

Διάφορες επιθέσεις (Miscellaneous). Υπάρχουν και άλλα είδη τρωτών σημείων που συνήθως έχουν ως αποτέλεσμα είτε καταρρεύσεις των διαδικασιών, απορύθμιση του πυρήνα, ή κολλήματα, όπως ανακατανομή των κενών δεικτών, διαίρεση με μηδενικά (CVE-2011-1012 και CVE-2010-4165), ατέρμονους βρόχους (CVE-2011-1083 και CVE-2010-1086) και αδιέξοδα (CVE-2.010 με 4.161).

Μία παρατήρηση από τον πίνακα 3.1 είναι ότι οι buffers και οι υπερχειλίσεις των ακεραίων είναι οι κορυφαίες απειλές για την ακεραιότητα του πυρήνα: το 78% των παραφθορών μνήμης προκαλούνται από αυτά τα δύο θέματα ευπάθειας. Αυτή η παρατήρηση είναι σύμφωνη με την έκθεση των Christiey και Martin. Ο πίνακας 3.2 δείχνει την κατανομή των τρωτών σημείων στο δενδρικό σύστημα του πυρήνα του Linux, δηλαδή ο κώδικας στατικά συνδέεται με την εικόνα του πυρήνα (core), τα προγράμματα οδήγησης συσκευών (drivers), τα πρωτόκολλα δικτύου (net), το σύστημα αρχείων (fs), και το υποσύστημα ήχου (sound). Μπορούμε να παρατηρήσουμε ότι ένα μη τετριμμένο τμήμα (1 = 3) των τρωτών σημείων βρίσκεται στο κέντρο του πυρήνα, ενώ 2 = 3 είναι loadable modules του πυρήνα. Λιγότερο από το 20% των τρωτών σημείων που εξετάσαμε βρίσκονται στα προγράμματα οδήγησης συσκευών.

Εξετάσαμε διάφορα εργαλεία ασφαλείας state-of-the-art του πυρήνα για να δούμε πόσες ευπάθειες μπορούν να αποτρέψουν. Ο πίνακας 3.3 συνοψίζει τα αποτελέσματα. Το υπόλοιπο κομμάτι τις παρουσίασης εξετάζει κάθε εργαλείο και τα τρωτά σημεία ή επιθέσεις που μπορεί να αποτρέψει. Η εξέταση μας επικεντρώνεται κυρίως σε εργαλεία που στοχεύουν τον πυρήνα, και βασίζονται στην κατανόηση αυτών των τεχνικών.

3.3.1 Εργαλεία κατά τον χρόνο εκτέλεσης

Software fault isolation. Το BGI είναι ένα εργαλείο απομόνωσης των modules του πυρήνα με υποστήριξη για την ελεγχόμενη ανταλλαγή μεταξύ του πυρήνα και των modules (**Chen et al**). Το BGI παρέχει μια λίστα ελέγχου πρόσβασης στη μνήμη (ACL) για κάθε module. Οι προγραμματιστές που χρησιμοποιούν το BGI εκχωρούν στην λίστα ACL, παρέχουν ή ανακαλούν τα προνόμια ενός module όπως αυτό επικαλείται διάφορες λειτουργίες του πυρήνα (Π.χ., την παροχή πρόσβασης κατά την κατανομή της μνήμης και την ανάκληση της πρόσβασης όταν απελευθερώνεται μνήμη). Με τον τρόπο αυτό, το εργαλείο BGI μπορεί να αποτρέψει ένα ευάλωτο module από το να αντικαταστήσει/ να γράψει επάνω από τη μνήμη πυρήνα (που δεν θα έπρεπε να έχει πρόσβαση), όπως τα double-free bugs και ορισμένες υπερχειλίσεις μνήμης (buffer overflows), αλλά επιτρέπει την πρόσβαση στη μνήμη του πυρήνα.
Ένα σημαντικό μειονέκτημα του BGI είναι ότι χειρίζεται μόνο τα τρωτά σημεία μέσα σε ένα module, καθώς και ορισμένες επιθέσεις που επιχειρούν να διαπεράσουν το όριο του εσφαλμένου (buggy) module. Όπως φαίνεται και στο σχήμα 3.2, 1 = 3 όλων των τρωτών σημείων βρίσκονται στον πυρήνα, και δεν θα πρέπει να χειρίζεται από το BGI. Επιπλέον, ορισμένα exploits συμβαίνουν εξ ολοκλήρου εντός των modules, και ως εκ τούτου, δεν θα πρέπει να διαχειρίζονται και αυτά από το BGI. Για παράδειγμα, αν υπάρχει μια ευπάθειας υπερχείλισης buffer σε ένα module του συστήματος αρχείων, ο εισβολέας θα μπορούσε να παρέμβει με εσωτερικά δεδομένα της κάρτας μνήμης, και έτσι να ξεγελάσει το module έτσι ώστε να εκτέλεση των κώδικα που ισχύει μόνο για τα setuid binaries και να αποκτήσει δικαιώματα διαχειριστή. Η απομόνωση των modules (το ένα από το άλλο), δεν θα έλυνε αυτό το πρόβλημα.

Code integrity. Το SecVisor επιβάλλει την ακεραιότητα του κώδικα πυρήνα. Ένα λεπτό επιβλέπων (hypervisor) στρώμα πιστοποιεί/επαληθεύει την ταυτότητα όλου του κώδικα πριν ο κωδικός αυτός επιτραπεί να εκτελεστεί στη λειτουργία πυρήνα. Το SecVisor είναι αποτελεσματικό στην πρόληψη επιθέσεων εγχύσεως κώδικα (code injection attacks). Ωστόσο, αν και μπορεί να προστατεύει έναντι κοινών exploits που αγνοούν την ύπαρξη του εργαλείου SecVisor, κακόβουλοι χρήστες που επιμένουν θα μπορούσαν ακόμη να εξαπολύσουν επιθέσεις για την παραφθορά σημαντικών δεδομένων του πυρήνα, ή να πάρουν τον έλεγχο της ροής ελέγχου στον υπάρχοντα κώδικα του πυρήνα. Έτσι, το SecVisor κάνει τα exploits πιο δύσκολο στο να εγγράψουν εσφαλμένα δεδομένα, αλλά δεν εμποδίζει έναν κακόβουλο χρήστη από την εκμετάλλευση των τρωτών σημείων, το οποίο είναι και ο λόγος που στο σχήμα 3.3 βάλαμε μηδενικά στη στήλη "SecVisor»

User-level drivers. Το εργαλείο SUD τρέχει τα προγράμματα οδήγησης συσκευών στο επίπεδο χρήστη και αποτρέπει τις ευπάθειες των οδηγών από το να επηρεάσουν τα υπόλοιπα τμήματα του πυρήνα. Αυτό μετατρέπει τις περισσότερες ευπάθειες των οδηγών σε μια DDoS επίθεση η οποία μπορεί να συντρίψει τους οδηγούς. Ένας ξεχωριστός μηχανισμός ανάκαμψης, όπως οι οδηγοί shadow, είναι απαραίτητος έτσι ώστε να μετριάσει τις επιπτώσεις από τη DDoS επίθεση. Ωστόσο θα πρέπει να σημειωθεί ότι μόνο ένα μικρό μέρος των τρωτών σημείων του πυρήνα προέρχονται από τα προγράμματα οδήγησης συσκευών (20%, πίνακας 3.2) και ότι οι οδηγοί στο επίπεδο χρήστη δεν αντιμετωπίζουν τα άλλα τρωτά σημεία.

Memory tagging. Τα συστήματα σήμανσης μνήμης, όπως το Raksha, μπορούν να ανιχνεύσουν πότε γίνεται κακή χρήση του κώδικα του πυρήνα από μη

αξιόπιστες εισόδους (από διεργασίες χρήστη ή από το δίκτυο), αποτρέποντας έναν κακόβουλο χρήστη από τις επιθέσεις τοποθέτησης εγχύσεως κώδικα ή άλλως να αναλάβει τους ελέγχους των ροών. Όπως και με το εργαλείο SecVisor, έτσι και με το Raksha αποτρέπονται ορισμένοι τύποι exploits που βασίζονται στο να πάρουν τον έλεγχο των ροών του πυρήνα.

Uninitialized memory tracking. Δύο συστήματα αντιμετωπίζουν τα προβλήματα της διαρροής ευαίσθητων πληροφοριών του πυρήνα δια μέσω τις απροετοίμαστης μνήμης. Το Kmemcheck είναι ένα εργαλείο εκτέλεσης στον πυρήνα του Linux που ανιχνεύει τα τρωτά σημεία της απροετοίμαστης μνήμης για ένα δεδομένο φόρτο εργασιών. Το Kmemcheck παρακολουθεί την κατάσταση προετοιμασίας του κάθε byte, με τη βοήθεια του κατανεμητή μνήμης του πυρήνα. Το Kmemcheck δεν μπορεί να ανιχνεύσει την ανάγνωση των δεδομένων που δεν έχουν προετοιμαστεί από τη στοίβα. Από την άλλη πλευρά το εργαλείο ασφαλείας deallocation (SD), μηδενίζει περιοδικά την στοίβα πυρήνα για να μείωση τις διαρροές πληροφοριών από τις απροετοίμαστες στοίβες. Κανένα από τα δύο αυτά εργαλεία δεν μπορεί να εγγυηθεί ότι θα βρει και θα κάνει πρόληψη όλων των σφαλμάτων αποκάλυψης πληροφοριών, αν και κάνουν το έργο των bugs ποιο δύσκολο σε σχέση με το αν δεν υπήρχαν καθόλου.

3.3.2 Εργαλεία κατά τον χρόνο μεταγλώττισης

Τα εργαλεία ανάλυσης κώδικα μπορούν να εντοπίσουν τα τρωτά σημεία, έτσι ώστε να διορθωθούν μια για πάντα από τους προγραμματιστές (**Chen et al**). Πολλά τέτοια εργαλεία έχουν χρησιμοποιηθεί για να εντοπίσουν και να διορθώσουν ένα ευρύ φάσμα προβλημάτων ασφαλείας στον πυρήνα του Linux. Ένας περιορισμός των περισσοτέρων στατικών εργαλείων ανάλυσης είναι ο μεγάλος αριθμός των λανθασμένων θετικών αποτελεσμάτων που επιστρέφουν. Έτσι, δεδομένου ότι δεν είναι παραγωγικό για τους προγραμματιστές να φιλτράρουν όλα αυτά τα λανθασμένα θετικά αποτελέσματα και να διορθώσουν όλα τα τρωτά σημεία, τα εργαλεία αυτά χρησιμοποιούνται σε μεγάλο βαθμό για τον εντοπισμό των διαφόρων bugs. Για να μειωθεί ο αριθμός των λανθασμένων θετικών αποτελεσμάτων, πολλά στατικά εργαλεία ανάλυσης απαιτούν από τους προγραμματιστές να παρέχουν σχόλια.

Μια ειδική κατηγορία των τρωτών σημείων που φαίνεται να είναι δύσκολο να εντοπιστεί χρησιμοποιώντας στατική ανάλυση είναι τα σημασιολογικά τρωτά σημεία, όπως τα bugs ελλιπών ελέγχων αδειών. Από την άλλη πλευρά, πολλά εργαλεία υπήρξαν αποτελεσματικά στην εξεύρεση πιθανών υπερχειλίσεων buffer, deadlock, και σφάλματα άπειρους βρόχων

Ο πίνακας 3.3 δείχνει ότι καμία από τις προηγούμενες έρευνες δεν πραγματεύεται τις ευπάθειες "ελλείπεις ελέγχους αδειών". Δυστυχώς, αυτές οι ευπάθειες μπορούν εύκολα να αξιοποιηθούν έτσι ώστε να αποκτήσουν προνόμια.

Ευπάθεια	BGI	SecVisor	SUD	Raksha	kmemcheck	SD
Missing pointer check	0	0	3	0	0	0
Missing permission check	0	0	1	0	0	0
Buffer overflow	1	0	1	0	0	0
Integer overflow (D)	0	0	1	0	0	0
Integer overflow (I)	0	0	1	0	0	0
Integer overflow (E)	0	0	3	0	0	0
Uninitialized data	0	0	13	0	1	23
Null dereference	11	0	3	0	0	0
Divide by zero	2	0	0	0	0	0
Infinite loop	0	0	1	0	0	0
Data race / deadlock	0	0	1	0	0	0
Memory mismanagement	1	0	1	0	0	0
Miscellaneous	0	0	0	0	0	0

Πίνακας 3.3 Αριθμός ευπαθειών που τα υφιστάμενα εργαλεία εκτέλεσης μπορούν να αποτρέψουν (πηγή: Chen et al)

3.4 Linux Security Module Framework

Τον Μάρτιο του 2001, η Εθνική Υπηρεσία Ασφαλείας (NSA) έδωσε μια παρουσίαση σχετικά με την ενίσχυση της ασφάλειας στο Linux (SELinux) στη Σύνοδο Κορυφής Linux Kernel 2.5 (**LSM Framework,** 2003). Το SELinux είναι μια υλοποίηση ευέλικτων ελέγχων μη διακριτικής πρόσβασης και ελέγχου στον πυρήνα του Linux. Αρκετά άλλα έργα ασφαλείας (π.χ. RSBAC, Μέδουσα) έχουν επίσης αναπτύξει ευέλικτες αρχιτεκτονικές ελέγχου πρόσβασης για τον πυρήνα του Linux, και διάφορα έργα έχουν αναπτύξει συγκεκριμένα μοντέλα ελέγχου πρόσβασης για το Linux (π.χ. LIDS, DTE, subdomain). Κάθε έργο έχει αναπτυχθεί και συντηρεί δικά του patch για τον πυρήνα έτσι ώστε να υποστηρίξει τις ανάγκες ασφαλείας.

Σε απάντηση στην παρουσίαση της NSA, ο Linus Torvalds έκανε μια σειρά από παρατηρήσεις και περιγράφει ένα πλαίσιο ασφάλειας που θα ήταν πρόθυμος να εντάξη στον πυρήνα του Linux. Περιέγραψε ένα framework που θα παρέχει μια σειρά από δίχτυα ασφαλείας για τον έλεγχο των εργασιών σε αντικείμενα του πυρήνα και μια σειρά από αδιαφανείς τομείς της ασφάλειας στις δομές δεδομένων του πυρήνα για τη διατήρηση των χαρακτηριστικών ασφαλείας. Το πλαίσιο αυτό θα μπορούσε στη συνέχεια να χρησιμοποιηθεί από modules του πυρήνα για την εφαρμογή κάθε επιθυμητού μοντέλου ασφάλειας.

Το Linux Security Modules (LSM) ξεκίνησε από την WireX. Το LSM είναι μια κοινή προσπάθεια ανάπτυξης από πολλά έργα ασφάλειας, συμπεριλαμβανομένων των Immunix, SELinux, SGI και Janus, και πολλά άλλα άτομα,

συμπεριλαμβανομένων των Greg Kroah-Hartman και James Morris, για να αναπτυχθεί ένα patch για τον πυρήνα του Linux που υλοποιεί το πλαίσιο αυτό.

Το Linux Security Modules (LSM) είναι ένα framework που επιτρέπει στον πυρήνα του Linux να υποστηρίζει μια ποικιλία μοντέλων ασφάλειας, αποφεύγοντας την ευνοιοκρατία προς οποιαδήποτε ενιαία εφαρμογή. Το πλαίσιο έχει λάβει άδεια σύμφωνα με τους όρους της GNU General Public License και είναι μέρος του πυρήνα του Linux από την έκδοση 2.6 και μετά.

Το framework LSM είναι κυρίως επικεντρωμένο στη στήριξη των μονάδων ελέγχου πρόσβασης και τον έλεγχο. Από μόνο του, το LSM framework δεν παρέχει καμία πρόσθετη ασφάλεια αλλά παρέχει την υποδομή για την υποστήριξη των μονάδων ασφαλείας.

Κεφάλαιο 4

4.1 Εισαγωγή

Στο κεφάλαιο αυτό θα προσπαθήσουμε να παραμετροποιήσουμε τα διάφορα συστήματα GNU/Linux εν προκειμένω να μας παρέχουν ένα συγκεκριμένο επίπεδο ασφάλειας. Μέσω του εργαλείου ανίχνευσης εισβολέων (Lynis) το οποίο είναι ανοικτού κώδικα θα ανακαλύψουμε τα τρωτά σημεία και εκείνες τις ρυθμίσεις που χρειάζεται το σύστημά μας έτσι ώστε να είναι περισσότερο ασφαλές. Επίσης πρίν από την παραμετροποίηση θα εγκαταστήσουμε βήμα βήμα το Slackware Linux προκειμένου να δούμε τις διάφορες παραμέτρους που μπορούμε να ρυθμίσουμε εξαρχής. Σε αντίθεση με άλλες διανομές του Linux (πχ. Ubuntu, linux mint κλπ) που δεν μας επιτρέπουν την παραμετροποίηση του συστήματος εξαρχής το Slackware παρέχει τα κατάλληλα εργαλεία για την ρύθμιση πολλών και σημαντικών στοιχείων του συστήματος.

Στις μέρες μας οι ευπάθειες και τα τρωτά σημεία έχουν πάρει αρκετά μεγάλες διαστάσεις. Χαρακτηριστικά παραδείγματα αποτελούν οι πάμπολλες κλοπές προσωπικών δεδομένων αλλά και τα διάφορα bugs που ανακαλύπτονται κατά καιρούς και εκμεταλλεύονται οι επίδοξοι εισβολείς.

Μόνο για το 2015 στον kernel του Linux ανακαλύφθηκαν 77 τρωτά σημεία σύμφωνα με την βάση δεδομένων του CVE και αν κάνουμε μία πρόσθεση όλων των ευπαθειών που έχουν ανακαλυφθεί από το 1999 έως και σήμερα θα δούμε ότι αυτές ξεπερνούν τις 1300. Αν και αυτός ο αριθμός ίσως να ακούγεται μικρός σε σχέση με άλλες εφαρμογές όπως το flash player της adobe αλλά και το Mac OS X της Apple, εντούτοις εκείνες οι ευπάθειες που δεν ανακαλύπτονται αλλά παραμένουν είναι πολύ περισσότερες. Εδώ θα πρέπει να τονίσουμε ότι όσα μέτρα ασφαλείας και αν λάβουμε πάντα θα υπάρχει κάποια κρυφή είσοδος με την οποία θα μπορεί να παραβιαστεί το σύστημά μας. Η παραμετροποίηση και η παρακάτω ανάλυση δεν αφορά τους προφανείς τρόπους με τους οποίους μπορούν οι επίδοξοι εισβολείς να κάνουν εξόρυξη πληροφοριών (πχ. Phishing μέσω email) αλλά εκείνους τους τρόπους με τους οποίους ένας χρήστης ή μία εταιρία μπορεί να προφυλάξει το σύστημα, από γνωστές ευπάθειες και μεθόδους που χρησιμοποιούνται και είναι διαδεδομένες.

4.2 Προετοιμασία εγκατάστασης συστήματος και έλεγχοι ασφαλείας

Για τις ανάγκες τις παρούσας εργασίας θα κάνουμε εγκατάσταση της διανομής του Slackware Linux η οποία αποτελεί και την παλαιότερη διανομή Linux η οποία είναι ακόμα ενεργή στο επίπεδο της συντήρησης. Η διανομή αυτή ενδείκνυται τόσο για χρήση ως Server όσο και προσωπική χρήση. Χάρη στην σταθερότητα και την ευελιξία που προσφέρει μπορεί να χαρακτηριστεί ως ένα ασφαλές λειτουργικό σύστημα με δυνατότητες μεγάλης παραμετροποίησης. Για την εγκατάσταση θα χρειαστούμε το iso αρχείο το οποίο μπορούμε να κατεβάσουμε είτε μέσω ftp, είτε μέσω αρχείου torrent.

Μία επισήμανση που θα πρέπει να γίνει είναι η εξής: κατά καιρούς οι διάφορες διανομές του Linux έχουν δεχτεί διάφορες επιθέσεις με αντικατάσταση των αρχείων iso και γενικότερα των αρχείων εγκατάστασης από τις επίσημες τοποθεσίες με άλλα κακόβουλα αρχεία. Η ασφαλέστερη μέθοδος που θα μπορούσε να χρησιμοποιηθεί είναι αυτή του torrent καθώς το αρχικό αρχείο διαμοιράζεται από πολλούς χρήστες και είναι σχεδόν αδύνατον να αντικατασταθεί από επίδοξους εισβολείς που θέλουν να το παραμετροποιήσουν προκειμένου να εξυπηρετήσει τα συμφέροντά τους. Επίσης μία άλλη μέθοδος που μπορούμε να εφαρμόσουμε αφού κατεβάσουμε το αρχείο εγκατάστασης είναι να ελέγξουμε την ακεραιότητα του μέσω των διαφόρων μεθόδων συναρτήσεων κατακερματισμού όπως η md5. Αυτό θα μπορούσαμε να πούμε πια ότι είναι επιβεβλημένο ειδικά αν πρόκειται να χρησιμοποιήσουμε το σύστημα ως Server.

Ας επιστρέψουμε όμως στην βασική εγκατάσταση του συστήματός μας. Το πρώτο βήμα που θα πρέπει να γίνει είναι να δημιουργήσουμε ένα bootable δίσκο (είτε σε μονάδα usb είτε σε dvd), προκειμένου να γίνει η εγκατάσταση του συστήματός μας μέσω αυτού. Υπάρχουν πολλά προγράμματα έτοιμα που μπορούμε να χρησιμοποιήσουμε για την δημιουργία του bootable δίσκου, όμως θα προτιμήσουμε την γραμμή εντολών του Unix και του Linux η οποία ναι μεν είναι πιο πολύπλοκη, όμως μας παρέχει περισσότερη αξιοπιστία σε σχέση με τα διάφορα προγράμματα που κυκλοφορούν. Γενικά η εγκατάσταση του συστήματός μας θα γίνει μέσω εντολών (CLI) και κάποιες ενέργειες μέσω γραφικού περιβάλλοντος.

Το πρώτο βήμα λοιπόν είναι να κατεβάσουμε το αρχείο iso από τον παρακάτω σύνδεσμο:

http://www.slackware.com/getslack/torrents.php

Θα εγκαταστήσουμε την x64 έκδοση. Αφού το κατέβασμα του αρχείου ολοκληρωθεί όπως είπαμε και προηγουμένως ελέγχουμε την ακεραιότητα του μέσω της συνάρτησης κατακερματισμού md5. Να σημειωθεί πως πριν την εγκατάσταση έχουμε ήδη και χρησιμοποιούμε ένα σύστημα GNU/Linux το οποίο είναι εγκατεστημένο σε ένα άλλο μηχάνημα από αυτό της εγκατάστασης. Για να ελέγξουμε την ακεραιότητα του αρχείου, βρίσκουμε το αρχείο *slackware64-14.1-install-dvd.iso.md5* που υπάρχει μέσα στον φάκελο του slackware που μόλις κατεβάσαμε. Μπορούμε να ανοίξουμε το αρχείο αυτό και να δούμε τον αρχικό παραγόμενο κωδικό *md5*. Για να μπορέσουμε να τον συγκρίνουμε με αυτόν από το αρχείο που μόλις κατεβάσαμε, ανοίγουμε το τερματικό και πληκτρολογούμε την εντολή su έτσι ώστε να αποκτήσουμε δικαιώματα διαχειριστή. Εφόσον πληκτρολογήσουμε την εντολή su θα μας ζητηθεί να δώσουμε και τον κωδικό μας προκειμένου να πιστοποιηθούμε από το σύστημα. Έπειτα μεταφέρουμε το αρχείο iso της εγκατάστασης στον φάκελο */home/diplomatiki/*, όπου home είναι ο αρχικός κατάλογος και το diplomatiki ο κατάλογος χρήστη. Το όνομα του καταλόγου χρήστη μπορεί να διαφέρει ανάλογα με το τι έχουμε επιλέξει κατά την εγκατάσταση. Ελέγχουμε την ακεραιότητα πληκτρολογώντας τις παρακάτω εντολές:

```
cd /home/diplomatiki/
md5sum slackware64-14.1-install-dvd.iso
```

Η πρώτη εντολή μας οδηγεί στο φάκελο χρήστη όπου έχουμε τοποθετήσει το αρχείο iso ενώ η δεύτερη εξάγει των κωδικό md5 του αρχείου. Να σημειωθεί επίσης πώς μπορούμε να δούμε τα αρχεία που περιέχονται στον φάκελο */home/diplomatiki/* μέσω της εντολής *ls*. Με αυτό τον τρόπο μπορούμε να συγκρίνουμε αν ο παραγόμενος κωδικός με εκείνον που υπάρχει στο αρχείο του φακέλου που κατεβάσαμε είναι ο ίδιος. Σε περίπτωση που δεν ταιριάζουν οι δύο κωδικοί είναι πολύ πιθανό το αρχείο μας είτε να μην έχει ολοκληρωθεί σωστά η μεταφορά του είτε να έχει αντικατασταθεί από άλλο αρχείο από εκείνο που αρχικά είχε διανεμηθεί από την κοινότητα του slackware. Σε κάθε περίπτωση είναι επιβεβλημένος ο έλεγχος της ακεραιότητας για κάθε τι σημαντικό που εγκαθίσταται στο σύστημά μας.

Έχοντας έτοιμο το αρχείο μας στο κατάλογο */home/diplomatiki/* από πριν μπορούμε τώρα να δημιουργήσουμε το bootable δίσκο μας για να εγκαταστήσουμε το σύστημά μας από αυτόν. Η εγκατάσταση του iso θα γίνει σε ένα usb δίσκο μιας και είναι αρκετά πιο γρήγορο σε σχέση με τα παραδοσιακά cd ή dvd. Για την δημιουργία εκτελούμε τις παρακάτω εντολές έχοντας τοποθετήσει και το usb σε μία ελεύθερη θύρα:

```
lsblk
dd  if=/home/diplomatiki/slackware64-14.1-install-dvd.iso
of=/dev/sdb
```

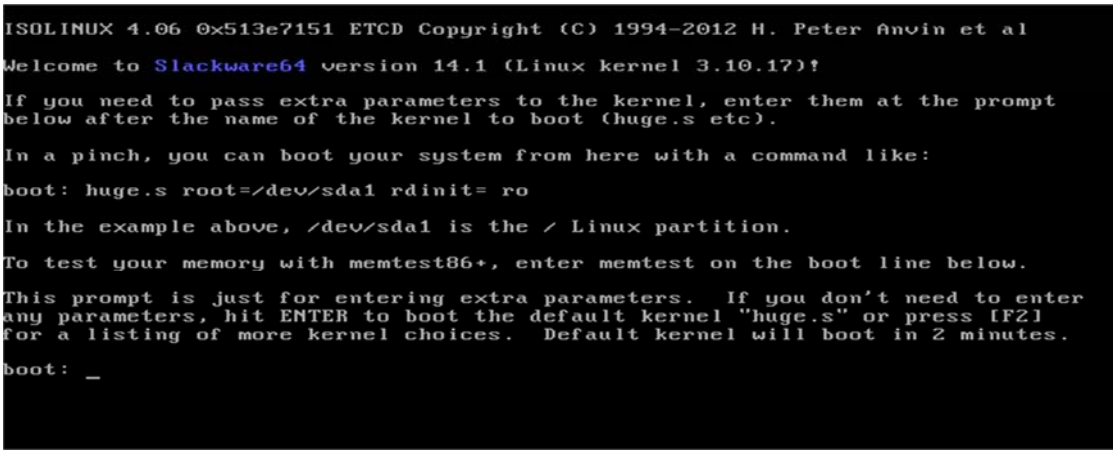
Δίνοντας την πρώτη εντολή μπορούμε να δούμε τις μονάδες δίσκων που υπάρχουν στο σύστημά μας. Συνήθως ο πρωταρχικός σκληρός μας δίσκος συμβολίζεται με το όνομα *sda* ενώ οι υπόλοιπες μονάδες ακολουθούν τη σειρά του αγγλικού αλφάβητου δηλαδή *sdb, sdc* κλπ

Εμείς θα πρέπει να εντοπίσουμε την μονάδα δίσκου του usb προκειμένου να εγκαταστήσουμε εκεί το αρχείο iso. Επίσης θα πρέπει να προσέξουμε να μην χρησιμοποιήσουμε κάποια κατάτμηση όπως *sdb1, sdb2* κλπ αλλά και να μην χρησιμοποιήσουμε τον πρωταρχικό μας σκληρό δίσκο (*sda*). Όλα τα δεδομένα στην μονάδα δίσκου usb θα διαγραφούν και θα αντικατασταθούν από εκείνα των αρχείων του συστήματος slackware. Με την εντολή dd λέμε ουσιαστικά στο σύστημά μας να δημιουργήσει έναν bootable δίσκο. Το */home/diplomatiki/slackware64-14.1-install-dvd.iso* είναι η διαδρομή που πρέπει να ακολουθήσει το σύστημά μας προκειμένου να εντοπίσει το αρχείο iso προς εγκατάσταση ενώ το *of=/dev/sdb* είναι η διαδρομή του usb δίσκου που όπως είχαμε επισημάνει και στις ενότητες για το σύστημα αρχείων, το dev είναι ο κατάλογος όπου υπάρχουν οι διάφορες διαδρομές των συσκευών συστήματος.

Μόλις δημιουργήσουμε το bootable usb μπορούμε να ξεκινήσουμε την εγκατάσταση. Αρχικά θα πρέπει να ρυθμίσουμε το BIOS ή το UEFI προκειμένου η προτεραιότητα των δίσκων από όπου μπορεί να φορτωθεί το λειτουργικό μας σύστημα να είναι πρώτα αυτή του usb και μετά όλων των υπολοίπων.

4.3 Εγκατάσταση του συστήματος

Κάνουμε επανεκκίνηση έχοντας τοποθετημένο το usb. Η πρώτη οθόνη που συναντάμε είναι η παρακάτω:



Σχήμα 4.1 Αρχική οθόνη εγκατάστασης Slackware

Στην οθόνη αυτή απλά θα πατήσουμε enter. Με αυτόν τον τρόπο θα φορτωθούν τα προκαθορισμένα αρχεία και ρυθμίσεις που χρειάζεται το σύστημά μας. Στη επόμενη οθόνη πληκτρολογούμε και πάλι enter για να χρησιμοποιήσουμε το πληκτρολόγιό μας στα αγγλικά.

Σχήμα 4.2 Οθόνη εγκατάστασης Slackware γλώσσα πληκτρολογίου

Στην παρακάτω οθόνη πληκτρολογούμε την εντολή `root` και πατάμε enter έτσι ώστε να αποκτήσουμε δικαιώματα διαχειριστή.

```
Welcome to the Slackware Linux installation disk! (version 14.1)

######  IMPORTANT!  READ THE INFORMATION BELOW CAREFULLY.  ######

- You will need one or more partitions of type 'Linux' prepared.  It is also
  recommended that you create a swap partition (type 'Linux swap') prior
  to installation.  For more information, run 'setup' and read the help file.

- If you're having problems that you think might be related to low memory, you
  can try activating a swap partition before you run setup.  After making a
  swap partition (type 82) with cfdisk or fdisk, activate it like this:
    mkswap /dev/<partition> ; swapon /dev/<partition>

- Once you have prepared the disk partitions for Linux, type 'setup' to begin
  the installation process.

- If you do not have a color monitor, type:  TERM=vt100
  before you start 'setup'.

You may now login as 'root'.

slackware login: root_
```

Σχήμα 4.3 Οθόνη εγκατάστασης Slackware root login

Έπειτα δίνουμε την εντολή `cfdisk`. Η εντολή αυτή θα μας οδηγήσει σε ένα μενού όπου μπορούμε να δημιουργήσουμε τις διάφορες κατατμήσεις στο σκληρό μας δίσκο που θα χρειαστούμε για την εγκατάσταση του συστήματός μας. Στο παρακάτω σχήμα βλέπουμε ότι ο δίσκος μας αποτελείτε από 16.8Gb ελεύθερου χώρου και ότι δεν υπάρχει καμία κατάτμηση.

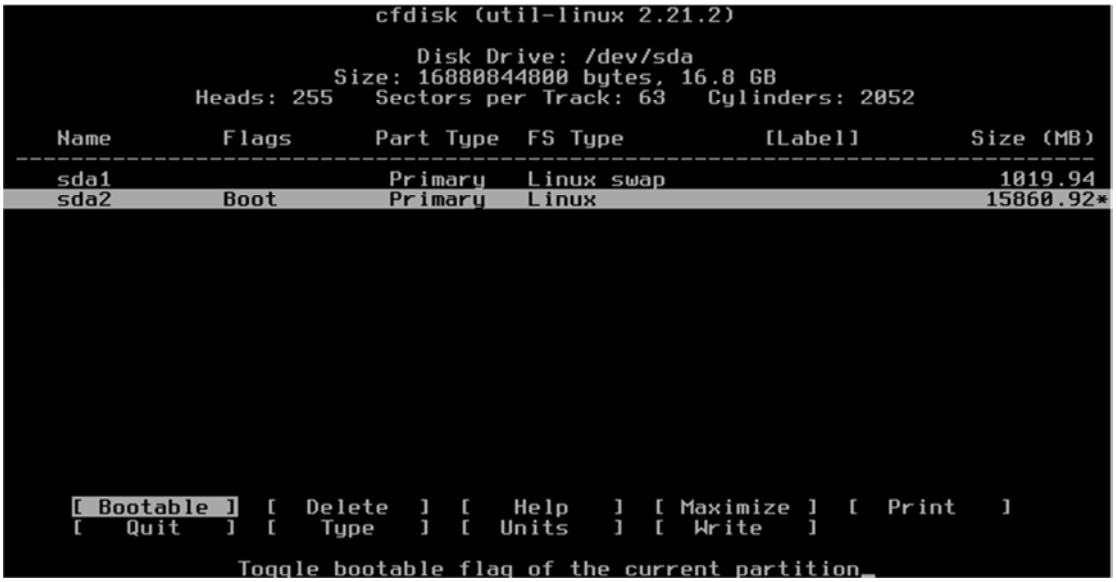
```
                         cfdisk (util-linux 2.21.2)

                            Disk Drive: /dev/sda
                      Size: 16880844800 bytes, 16.8 GB
             Heads: 255   Sectors per Track: 63   Cylinders: 2052

   Name        Flags      Part Type  FS Type          [Label]        Size (MB)
 ------------------------------------------------------------------------------
                           Pri/Log   Free Space                      16880.85*

     [   Help   ]  [   New    ]  [  Print   ]  [   Quit   ]  [  Units   ]
     [  Write   ]

                        Create new partition from free space.
```

Σχήμα 4.4 Οθόνη εγκατάστασης Slackware cfdisk partitions (a)

Επιλέγουμε το `New` και στη συνέχεια το `primary` προκειμένου να δημιουργήσουμε τις κατατμήσεις που απαιτούνται. Η πρώτη κατάτμηση θα είναι η `sda1` και εκεί θα βρίσκεται το `swap`. Το `swap` ή η κατάτμηση `swap` δρα ως μία επιπλέον πηγή, όταν η μνήμη RAM δεν είναι επαρκείς. Μπορούμε επίσης να μην δημιουργήσουμε καθόλου την κατάτμηση `swap`. Αυτό εξαρτάτε από την χρήση που κάνουμε και από το σύστημά μας. Γενικά ένα σύστημα πάντα χρειάζεται αρκετή

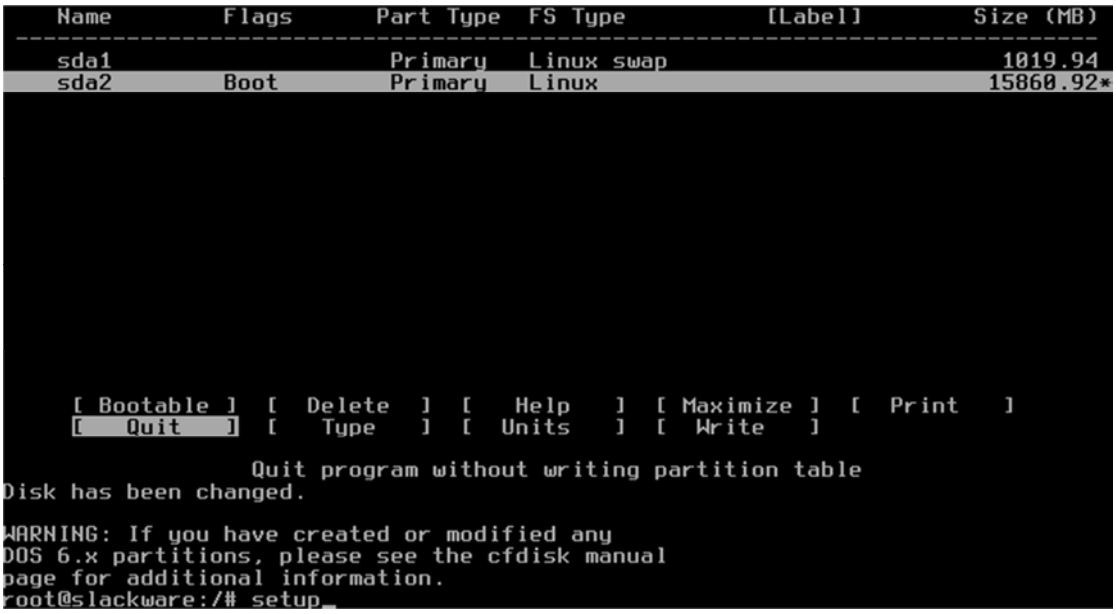
μνήμη ram για την απρόσκοπτη λειτουργία του, όμως τα σύγχρονα συστήματα είναι εφοδιασμένα με αρκετά Gb μνήμης ram οπότε πολλές φορές το swap δεν εξυπηρετεί. Τουναντίον στους σύγχρονους σκληρούς δίσκους της νέας τεχνολογίας SSD επιβαρύνεται η διάρκεια ζωής του δίσκου καθώς οι SSD έχουν συγκεκριμένο αριθμό εγγραφών, πριν τερματιστεί η λειτουργία τους.

Η δεύτερη κατάτμηση είναι η *sd2* στην οποία θα εγγράψουμε το βασικό μας σύστημα και από την οποία θα γίνεται το boot του λειτουργικού μας συστήματος.



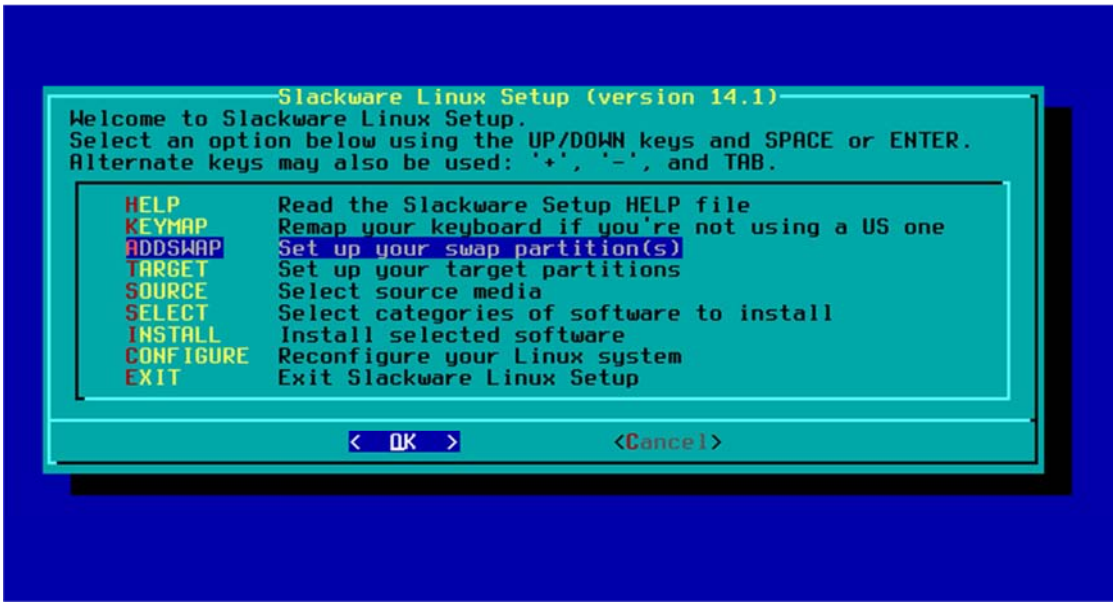
Σχήμα 4.5 Οθόνη εγκατάστασης Slackware cfdisk partitions (b)

Επιλέξαμε το μέγεθος της κατάτμησης swap να είναι περίπου 1Gb και το υπόλοιπο του δίσκου να κατανεμηθεί στο βασικό μας σύστημα. Αφού τα δημιουργήσουμε επιλέγουμε το write και στην συνέχεια yes έτσι ώστε να εγγραφούν/δημιουργηθούν οι κατατμήσεις στο δίσκο μας. Έπειτα επιλέγουμε το quit. Το επόμενο βήμα είναι η εγκατάσταση. Πληκτρολογούμε το setup για να ξεκινήσουμε την διαδικασία.



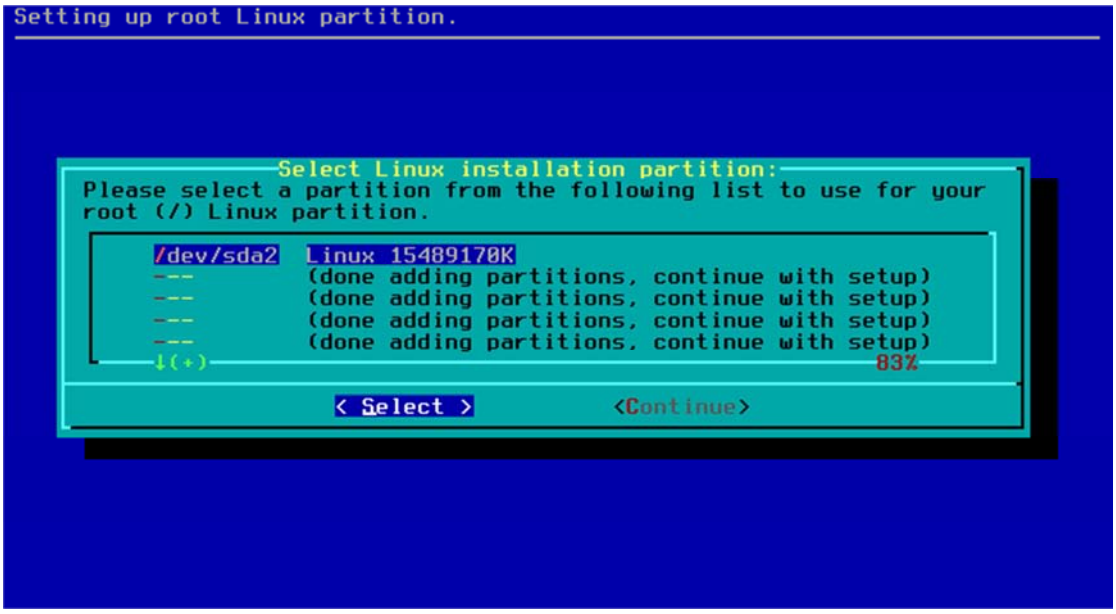
Σχήμα 4.6 Οθόνη εγκατάστασης Slackware cfdisk partitions (c)

Στην παρακάτω οθόνη που εμφανίζεται επιλέγουμε το *ADDSWAP* έτσι ώστε να δώσουμε στο σύστημα την κατάτμηση του swap που δημιουργήσαμε προηγουμένως. Επιλέγουμε το swap και ok. Στη συνέχεια μας ρωτάει το σύστημα αν θέλουμε να ελέγξουμε τον σκληρό μας δίσκο για κατεστραμμένους τομείς. Παραλείπουμε το συγκεκριμένο βήμα μιας και γνωρίζουμε πως ο σκληρός μας δίσκος δεν έχει κατεστραμένους τομείς.



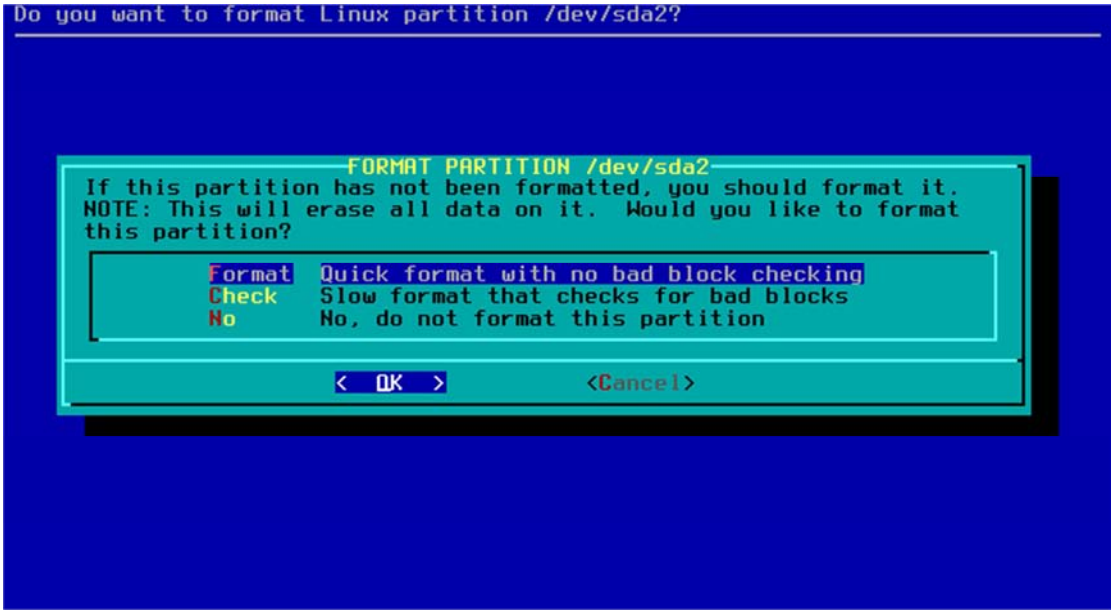
Σχήμα 4.7 Οθόνη εγκατάστασης Slackware addswap

Η επόμενη οθόνη μας ρωτάει για το που θα εγκαταστήσουμε το σύστημά μας, και επιλέγουμε την κατάτμηση *sda2* που και αυτή την είχαμε δημιουργήσει προηγουμένως.



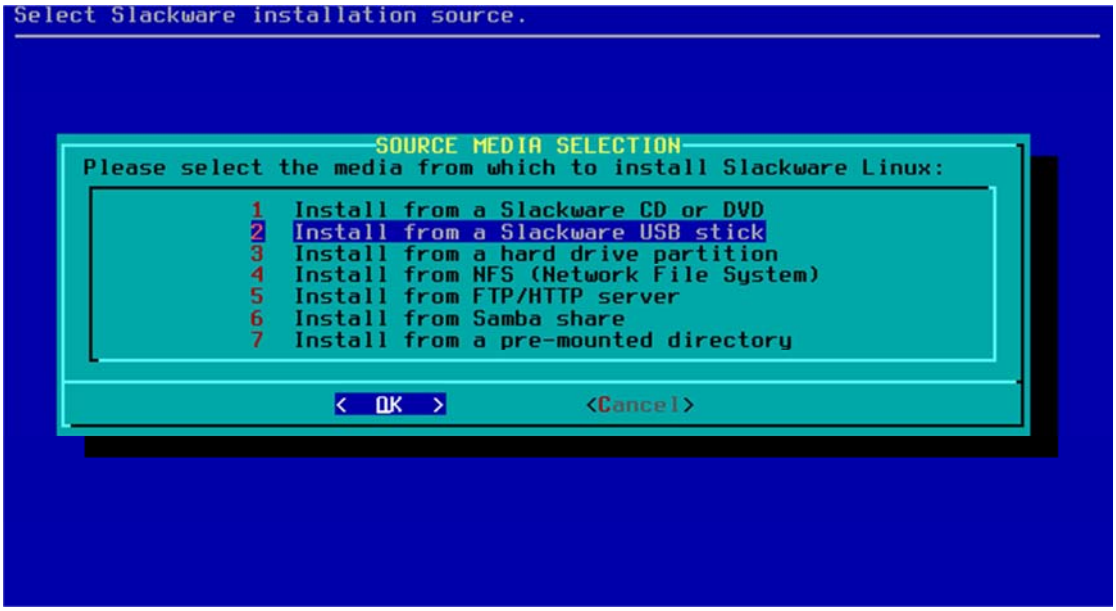
Σχήμα 4.8 Οθόνη εγκατάστασης Slackware installation partition

Έπειτα το σύστημα μας ρωτάει αν θέλουμε να διαμορφώσουμε τον σκληρό μας δίσκο. Επιλέγουμε όπως φαίνεται στην παρακάτω οθόνη και μετά το format του συστήματος αρχείων το οποίο θα είναι το *ext4 journaling*. Μπορούμε να επιλέξουμε οποιοδήποτε διαμόρφωση συστήματος αρχείων, όμως ενδείκνυται το *ext4 journaling* για τους λόγους που έχουν εξηγηθεί σε προηγούμενα κεφάλαια.



Σχήμα 4.9 Οθόνη εγκατάστασης Slackware format partition

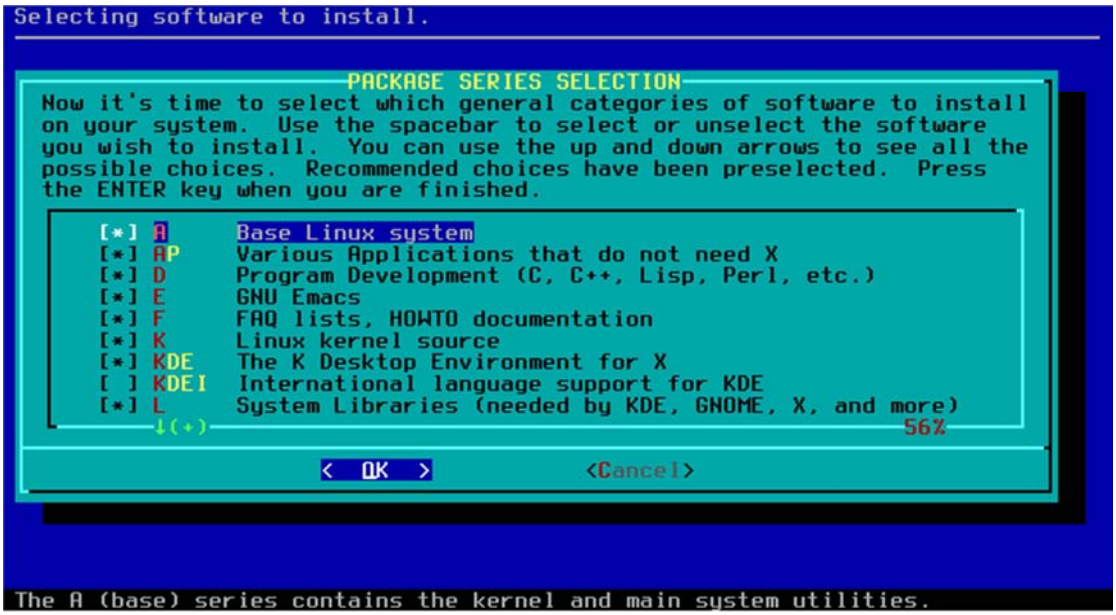
Επόμενο βήμα επιλέγουμε το ok και έπειτα όπως φαίνετε και στο παρακάτω σχήμα επιλέγουμε `install from USB stick` το οποίο το είχαμε δημιουργήσει στα αρχικά βήματα.



Σχήμα 4.10 Οθόνη εγκατάστασης Slackware source media selection

Θα μας ρωτήσει αν θέλουμε να εντοπίσει τα αρχεία εγκατάστασης αυτόματα ή να του υποδείξουμε την διαδρομή. Εμείς επιλέγουμε τον αυτόματο τρόπο.

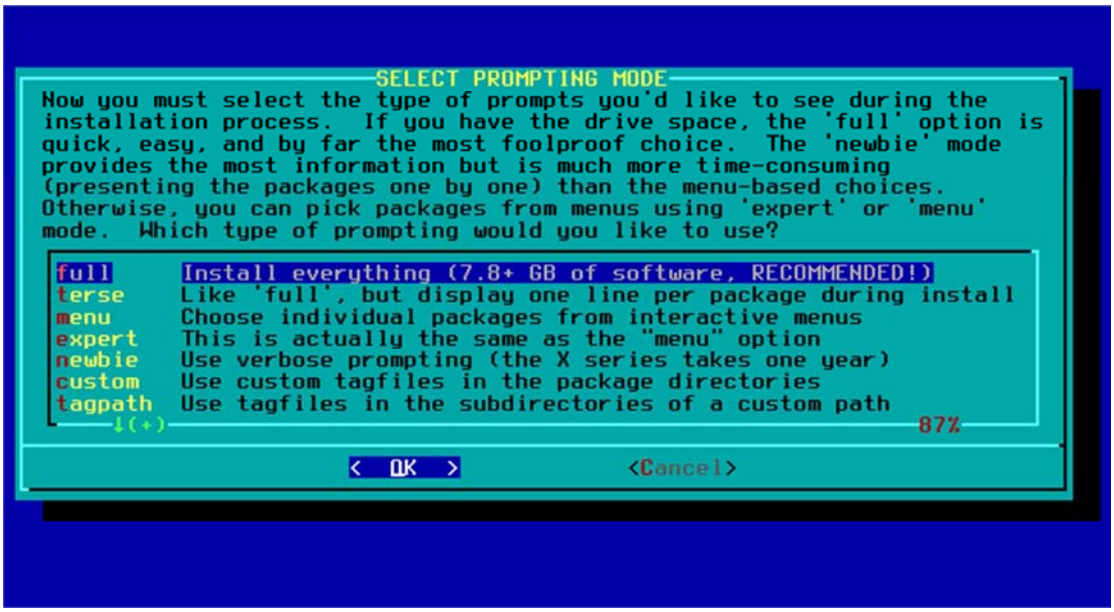
Στην επόμενη (παρακάτω) οθόνη θα πρέπει να επιλέξουμε τις υπηρεσίες και τα προγράμματα/εφαρμογές που θέλουμε να εγκατασταθούν μαζί με τα βασικά συστατικά στοιχεία που αποτελούν την διανομή.



Σχήμα 4.11 Οθόνη εγκατάστασης Slackware package series selection

Εάν επιλέξουμε μόνο τα δύο πρώτα το σύστημά μας θα αποτελείτε μόνο από τον πυρήνα του Linux και μερικά βοηθητικά προγράμματα που δεν χρειάζονται γραφικό περιβάλλον. Θα προτιμήσουμε τα προκαθορισμένα με το γραφικό περιβάλλον KDE. Επόμενο βήμα επιλέγουμε *install everything*. Σε αυτό το στάδιο θα μπορούσαμε να επιλέξουμε ποιες υπηρεσίες και προγράμματα/εφαρμογές

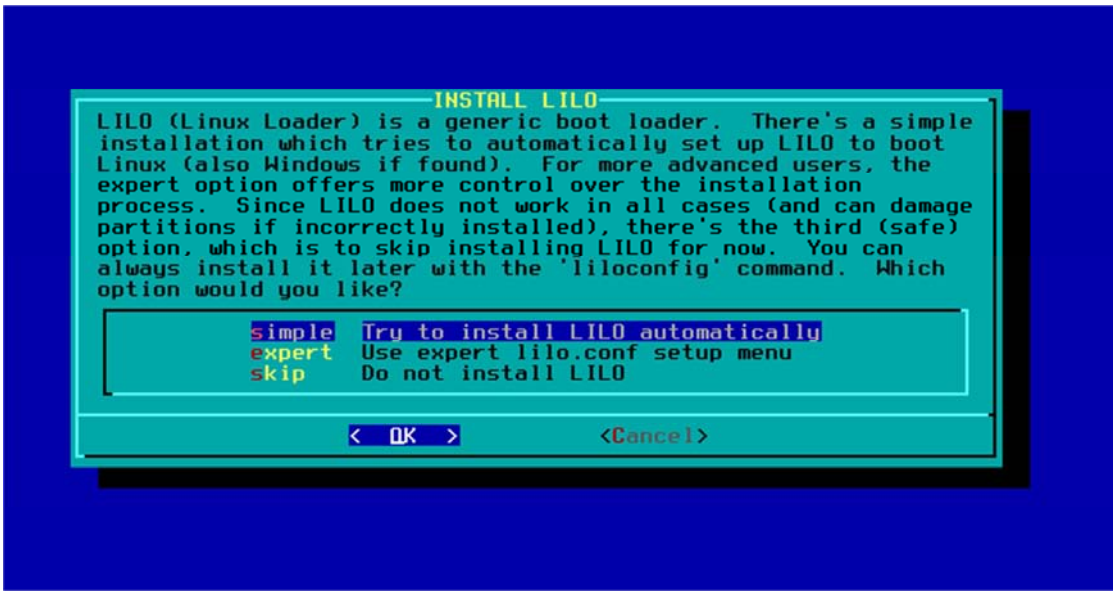
θέλουμε να έχει το σύστημά μας μέσα από την επιλογή menu. Αυτός ο τρόπος μας δίνει τον πλήρη έλεγχο για το τι θα εγκατασταθεί στο σύστημά μας. Θα πρέπει όμως να προσέξουμε οι εξαρτήσεις των πακέτων που θα επιλεγούν να είναι οι σωστές καθώς υπάρχει μεγάλη πιθανότητα το σύστημά μας να μην λειτουργεί σωστά. Αυτό το στάδιο είναι σημαντικό και μπορεί να χρησιμοποιηθεί για να διαμορφώσουμε ένα σύστημα ελαφρύ και χωρίς περιττά στοιχεία τα οποία δεν θα χρησιμοποιήσουμε ποτέ. Μπορούμε να αποεπιλέξουμε τα όποια στοιχεία πηγαίνοντας στο καθένα που επιθυμούμαι και πατώντας το πλήκτρο backspace



Σχήμα 4.12 Οθόνη εγκατάστασης Slackware prompting mode

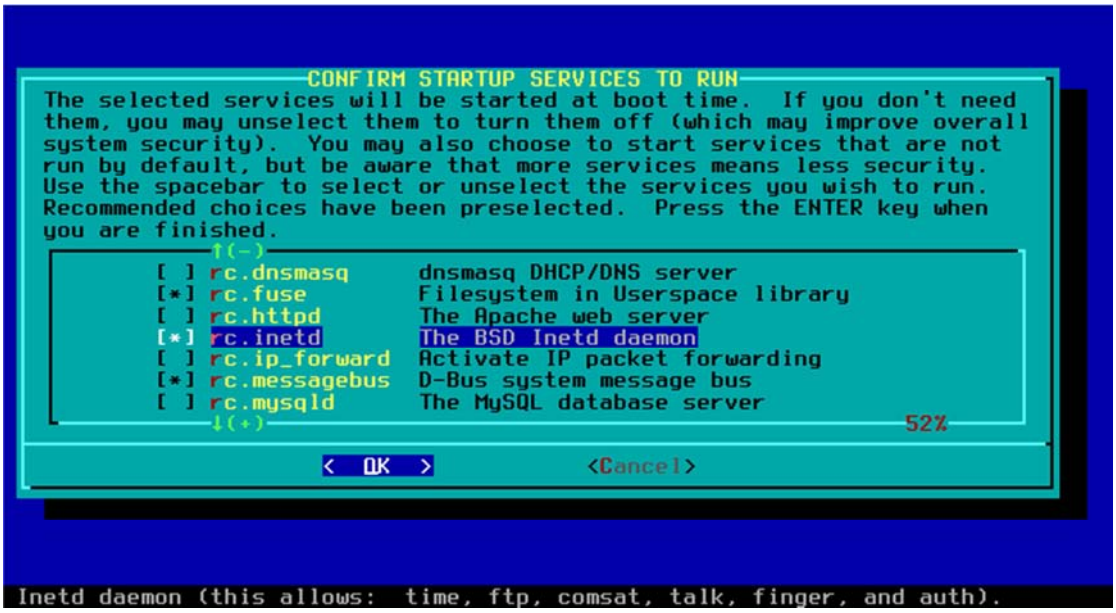
Στη συνέχεια γίνεται η αυτόματη εγκατάσταση των στοιχείων που έχουμε επιλέξει καθώς και του πυρήνα και των άλλων απαραίτητων στοιχείων.

Η επόμενη οθόνη μας ρωτάει για την εγκατάσταση του εκκινητή LILO (ενότητα 1.3.1 και 3.1.1). Επιλέγουμε το *simple*, στην συνέχεια `standard` και enter. Στην επόμενη οθόνη επιλέγουμε no και έπειτα λέμε στο σύστημά μας να εγκαταστήσει το LILO στο `master boot`.



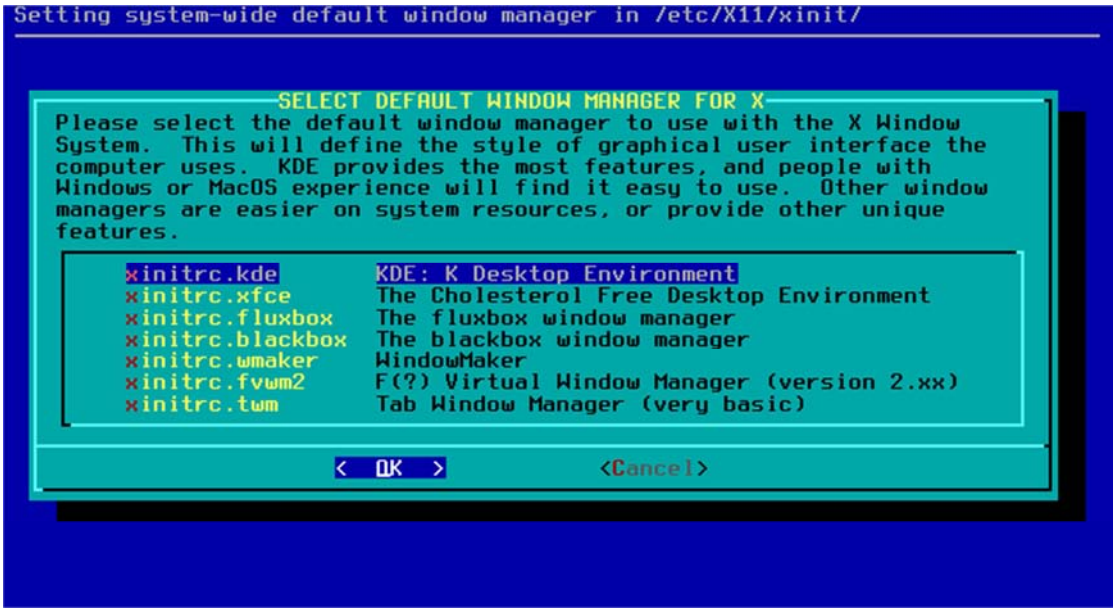
Σχήμα 4.13 Οθόνη εγκατάστασης Slackware lilo install

Έπειτα επιλέγουμε να εκκινείτε το LILO σε κάθε εκκίνηση του συστήματος και στη συνέχεια διαμορφώνουμε τις ρυθμίσεις δικτύου. Επιλέγουμε το `NetworkManager` για αυτόματη εγκατάσταση των ρυθμίσεων και στη συνέχεια enter. Στην παρακάτω οθόνη μπορούμε να δούμε τις υπηρεσίες/δαίμονες δικτύου που επιθυμούμαι να εγκατασταθούν. Σημαντική όπως έχουμε εξηγήσεις και στο κεφάλαιο 2 η `rc.inetd`, καθώς επίσης και οι δαίμονες `rc.fuse` για το σύστημα αρχείων, η `rc.syslog` και η `rc.sshd`. Μπορούμε επίσης αν χρησιμοποιούμε την mysql να ενεργοποιήσουμε και να εγκαταστήσουμε την υπηρεσία/δαίμονα `rc.mysqid`. Προχωρούμε με τις προκαθορισμένες υπηρεσίες/δαίμονες.



Σχήμα 4.14 Οθόνη εγκατάστασης Slackware start up services

Στη συνέχεια θα επιλέξουμε την ζώνη ώρας που επιθυμούμαι και έπειτα θα μας ζητηθεί να επιλέξουμε το γραφικό περιβάλλον της αρεσκείας μας. Στο παρακάτω σχήμα φαίνονται οι επιλογές των GUIs που είναι διαθέσιμα.



Σχήμα 4.15 Οθόνη εγκατάστασης Slackware window manager for x

Επιλέγουμε το KDE καθώς είναι ένα εξαιρετικό περιβάλλον. Να σημειωθεί πώς το KDE είναι και προσωπική επιλογή του Linus Torvalds.

Στην επόμενη οθόνη δημιουργούμε/εισάγουμε τον κωδικό διαχειριστή ο οποίος θα μας ζητείτε κάθε φορά που κάνουμε είσοδο στο σύστημά μας αλλά και όταν πρόκειται να εκτελέσουμε σημαντικές εργασίες. Η εγκατάστασή μας έχει ολοκληρωθεί και μπορούμε να χρησιμοποιήσουμε το σύστημά μας. Με την εντολή `reboot` μπορούμε να κάνουμε επανεκκίνηση και η πρώτη οθόνη είναι αυτή που φαίνεται στο παρακάτω σχήμα.

Σχήμα 4.16 Αρχική οθόνη εκκίνησης Slackware

Πατάμε enter για να εκκινήσουμε το σύστημα. Εκκινούνται οι διάφορες υπηρεσίες, οδηγοί, προγράμματα και έπειτα μας ζητείται η είσοδος στο σύστημα. Επειδή δεν έχουμε δημιουργήσει κάποιον χρήστη ο μοναδικός που υπάρχει είναι ο *root*. Εισάγουμε το *root* και έπειτα τον κωδικό που είχαμε δώσει σε προηγούμενο βήμα.

```
¦                  ¦
+------------------+
Starting ACPI daemon:  /usr/sbin/acpid
Updating MIME database:  /usr/bin/update-mime-database /usr/share/mime &
Starting system message bus:  /usr/bin/dbus-uuidgen --ensure ; /usr/bin/dbus-dae
mon --system
Starting ConsoleKit daemon:  /usr/sbin/console-kit-daemon
Starting NetworkManager daemon:  /usr/sbin/NetworkManager
Updating gtk.immodules for x86_64-slackware-linux:
  /usr/bin/gtk-query-immodules-2.0-64 --update-cache
Updating gtk.immodules for x86_64-slackware-linux:
  /usr/bin/gtk-query-immodules-3.0-64 --update-cache
Updating gdk-pixbuf.loaders for x86_64-slackware-linux:
  /usr/bin/gdk-pixbuf-query-loaders-64 --update-cache
Updating pango.modules for x86_64-slackware-linux:
  /usr/bin/pango-querymodules-64 \
  > /etc/pango/x86_64-slackware-linux/pango.modules
Setting default ALSA mixer settings.
Storing default ALSA mixer settings:  /usr/sbin/alsactl store
Starting gpm:  /usr/sbin/gpm -m /dev/mouse -t imps2


Welcome to Linux 3.10.17 (tty1)

diplomatiki login:
```

Σχήμα 4.17 Αρχική οθόνη εκκίνησης Slackware login

Για να δημιουργήσουμε κάποιον χρήστη αρκεί να δώσουμε την εντολή *adduser* και έπειτα το όνομα του χρήστη που θέλουμε. Δίνουμε λοιπόν την εντολή:

adduser diplomatiki

και στη συνέχεια enter και πάλι enter. Στη συνέχεια μας ρωτάει για τον φάκελο */home* και ποιος θέλουμε να είναι. Εμείς επιλέγουμε τον προκαθορισμένο δηλαδή των */home/diplomatiki/* και έπειτα μας ρωτάει για το κέλυφος που θέλουμε να χρησιμοποιήσουμε. Επιλέγουμε το *bash*. Δεν δίνουμε επιπλέον στοιχεία όπως ημερομηνία λήξης για τον λογαριασμό, αν και είναι χρήσιμο σε περιπτώσεις εταιρείας που θέλουμε οι εργαζόμενοι να έχουν περιορισμένα δικαιώματα η και ημερομηνία λήξης στους διάφορους λογαριασμούς. Τέλος δίνουμε τον κωδικό πρόσβασης για τον χρήστη diplomatiki. Δίνοντας την εντολή *exit* αποσυνδεόμαστε από τον χρήστη root και έπειτα μπορούμε να συνδεθούμε ως χρήστης diplomatiki, με τα στοιχεία εισόδου που δηλώσαμε προηγουμένως. Για να εκκινήσουμε το γραφικό περιβάλλον, δίνουμε την εντολή *startx*. Το παρακάτω σχήμα φαίνεται η εκκίνηση του γραφικού περιβάλλοντος KDE.

Σχήμα 4.18 Αρχική οθόνη εκκίνησης Slackware KDE GUI

4.4 Ρυθμίσεις ασφαλείας και άλλες ρυθμίσεις στο slackware

Οι αρχικές ρυθμίσεις που θα πρέπει να γίνουν προκειμένου να έχουμε ένα ασφαλές λειτουργικό σύστημα αφορούν την ενημέρωση των παλαιών πακέτων λογισμικού που υπάρχουν στο σύστημά μας.

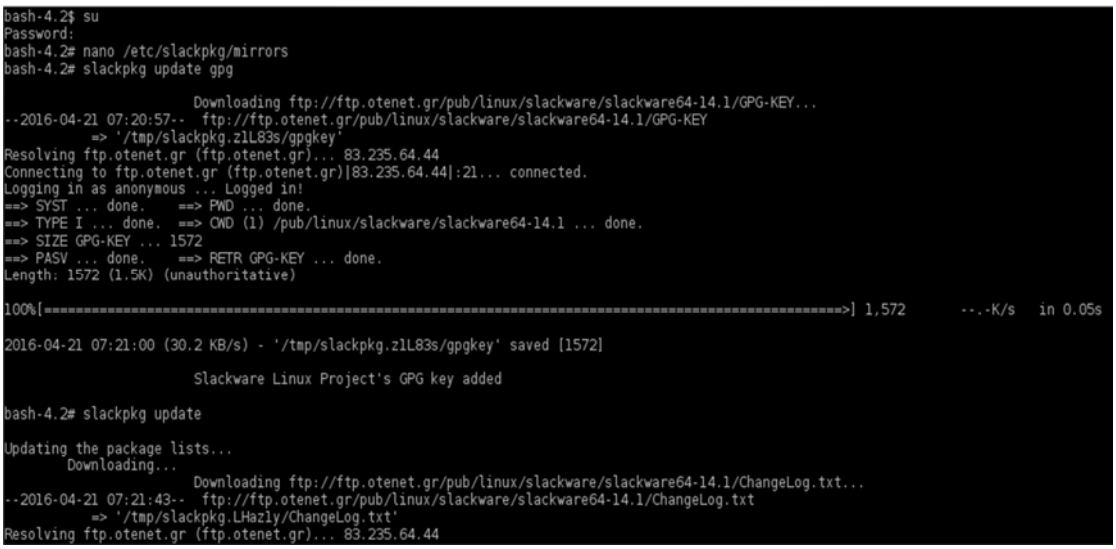
Πρώτο βήμα είναι να επιλέξουμε έναν mirror (ουσιαστικά πρόκειται για ένα ftp) από όπου μπορούμε να έχουμε αυτόματες ενημερώσεις. Το συγκεκριμένο βήμα κρίνετε απαραίτητο αφού σε διαφορετική περίπτωση το σύστημά μας θα παραμείνει χωρίς ενημερώσεις ασφαλείας ή άλλες ενημερώσεις λογισμικού. Αυτό μπορεί να πραγματοποιηθεί μέσω του τερματικού bash με τις παρακάτω εντολές (αφού αποκτήσουμε δικαιώματα διαχειριστή μέσω της εντολής *su*):

```
nano /etc/slackpkg/mirrors
```

Σημαντική σημείωση: Ποτέ δεν κάνουμε είσοδο στο σύστημά μας με δικαιώματα διαχειριστή.

Το nano είναι ένας editor ο οποίος θα μας βοηθήσει στην επεξεργασία τις λίστας των mirrors. Με την παραπάνω εντολή ουσιαστικά ανοίγουμε προς επεξεργασία το αρχείο mirrors που βρίσκεται στον φάκελο etc και στον υποφάκελο *slackpkg*. Για να επιλέξουμε κάποιον mirror αρκεί να διαγράψουμε την δίεση που υπάρχει μπροστά από τον mirror τον οποίο επιθυμούμαι να μας παρέχει ενημερώσεις. Καλό είναι να επιλέξουμε κάποιον mirror ο οποίος είναι στην Ελλάδα. Στη συνέχεια θα πρέπει να κάνουμε ενημέρωση (update) το GPG[1] λογισμικό κρυπτογράφησης και έπειτα ενημέρωση του συστήματος με τις παρακάτω εντολές:

```
slackpkg update gpg
slackpkg update
```



[1]Το GNU Privacy Guard (GnuPG or GPG) είναι ένα υποκατάστασστο/αντικαταστάτης λογισμικού κρυπτογράφισης του PGP λογισμικού της Symantec. Το GnuPG είναι ένα πρόγραμμα λογισμικού υβριδικής-κρυπτογράφησης επειδή χρησιμοποιεί ένα συνδυασμό συμβατικού συμμετρικού κλειδιού κρυπτογράφησης (για ταχύτητα) και την κρυπτογραφία δημόσιου κλειδιού για την ασφαλή ανταλλαγή κλειδιού.

Στη συνέχεια θα πρέπει να κάνουμε εγκατάσταση όλων των προγραμμάτων που έχουν ενημερώσεις, χρησιμοποιώντας τις παρακάτω εντολές:

```
slackpkg install-new
slackpkg upgrade-all
```

Αν δεν επιθυμούμαι την ενημέρωση κάποιου προγράμματος μπορούμε να το παραλείψουμε, βρίσκοντάς το και πατώντας το πλήκτρο του backspace το οποίο το αποεπιλέγη. Η συγκεκριμένη ενέργεια δεν είναι ασφαλής καθώς μπορεί αν δεν γνωρίζουμε να παραλείψουμε κάποια σημαντική ενημέρωση ασφαλείας. Στο τέλος είναι σημαντικό να καθαρίσουμε το σύστημά μας από τα πακέτα που έχουν απομείνει μετά την εγκατάσταση καθώς και από τα πακέτα που δεν ήταν εγκατεστημένα στην αρχική εγκατάσταση του συστήματος, με την παρακάτω εντολή:

```
slackpkg clean-system
```

Να σημειωθεί πώς τα περισσότερα εργαλεία/εφαρμογές που μας είναι απαραίτητα μας παρέχονται από την αρχική εγκατάσταση του slackware, μπορούμε φυσικά να εγκαταστήσουμε και επιπρόσθετα πακέτα από αξιόπιστες πηγές των κοινοτήτων slackware. Επίσης σημαντικό είναι οι ενημερώσεις που αφορούν τον kernel, τα στοιχεία και τις εξαρτήσεις αυτού να πραγματοποιούνται οπωσδήποτε διότι αποτελούν κρίσιμα στοιχεία του λειτουργικού μας συστήματος. Άλλα στοιχεία τα οποία είναι απαραίτητα να ενημερωθούν είναι η εφαρμογή του browser και το ssh.

Ο πυρήνας (kernel) στο slackware 14.1 είναι ο 3.10.x ο οποίος είναι μακράς υποστήριξης (LTS) δηλαδή θα παίρνει ενημερώσεις ασφαλείας για ένα μεγάλο χρονικό διάστημα. Το διάστημα αυτό έχει οριστεί να είναι μέχρι τον Οκτώβριο του 2017. Αν αναλογιστούμε ότι η έκδοση 14.1 του slackware είναι ενεργή από το 2013 αυτό συνεπάγεται υποστήριξη ασφαλείας για 4 με 5 χρόνια για το σύστημά μας. Η επερχόμενη έκδοση 14.2 του slackware θα υποστηρίζει τον πυρήνα 4.4.x ο οποίος είναι και αυτός μακράς υποστήριξης (LTS).

4.4.1 Παρακολούθηση ενημερώσεων ασφαλείας

Όταν ένα πρόβλημα ασφαλείας επηρεάζει το σύστημα του Slackware, αποστέλλεται ένα μήνυμα ηλεκτρονικού ταχυδρομείου σε όλους τους εγγεγραμμένους στη λίστα *slackware-security@slackware.com*. Στέλνονται αναφορές για ευπάθειες σε οποιοδήποτε κομμάτι του Slackware, εκτός από το λογισμικό στο /extra ή στο /pasture. Αυτές οι ανακοινώσεις ασφαλείας περιέχουνε λεπτομέρειες στο πως να προμηθευτούμε τις ανανεωμένες εκδόσεις των πακέτων για το Slackware ή διορθώσεις, αν υπάρχουν. Καλό είναι να παρακολουθούμε την λίστα αυτή με τις ευπάθειες που επηρεάζουν το λειτουργικό μας σύστημα. Επίσης όπως εξηγήσαμε και στην ενότητα 4.4 μπορούμε να εφαρμόζουμε το update και το upgrade σε καθημερινή βάση για να ενημερώνεται το σύστημά μας. Άλλος ένας τρόπος για ενημέρωση είναι μέσω της επίσημης ιστοσελίδας του slackware στην παρακάτω διεύθυνση:

http://www.slackware.com/lists/archive/

Όποτε βγαίνουν ανανεωμένα πακέτα για μια έκδοση του Slackware τοποθετούνται στο φάκελο /patches. Το πλήρες μονοπάτι για αυτές τις διορθώσεις εξαρτάται από το mirror που χρησιμοποιούμε, αλλά θα έχει την μορφή /path/to/slackware-x.x/patches/. Πριν εγκαταστήσουμε αυτά τα πακέτα, είναι μια καλή ιδέα να επιβεβαιώνουμε το md5sum του πακέτου όπως ακριβώς κάναμε και κατά την εγκατάσταση του λειτουργικού μας συστήματος.

4.4.2 Εγκατάσταση, απεγκατάσταση και ενημέρωση πακέτων

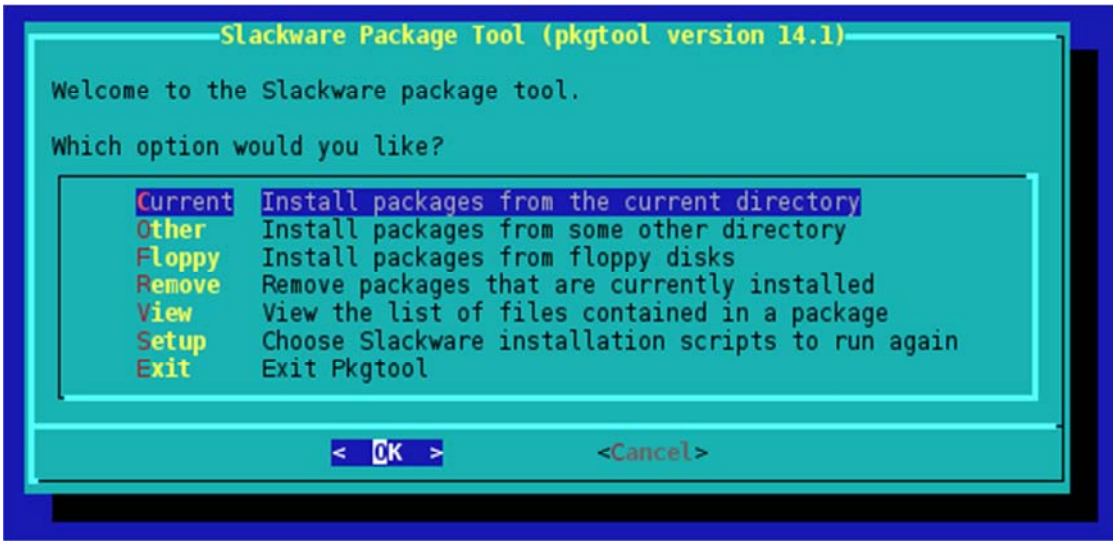
Στην περίπτωση που θέλουμε να κάνουμε εγκατάσταση ενημέρωσης ενός πακέτου χειροκίνητα μπορούμε να το κάνουμε μέσω τις εντολής:

```
upgradepkg packagename
```

όπου packagename το όνομα του πακέτου προς εγκατάσταση. Αν το νέο πακέτο δεν έχει το ίδιο όνομα με το παλαιό τότε θα πρέπει να χρησιμοποιήσουμε την εντολή:

```
upgradepkg oldpackagename%newpackagename
```

Ένα εργαλείο που παρέχεται μαζί με την εγκατάσταση του slackware είναι το `pkgtool`. Το εργαλείο αυτό μας δίνει την δυνατότητα να εγκαταστήσουμε ή να αφαιρέσουμε πακέτα από το σύστημά μας. Δίνοντας την εντολή αυτή στο τερματικό του συστήματός μας θα δούμε την παρακάτω οθόνη:



Σχήμα 4.19 Αρχική οθόνη pkgtool

Οι επιλογές που μας δίνονται είναι η εγκατάσταση από το κατάλογο /home/diplomatiki (current) που είχαμε ορίσει κατά την εγκατάσταση του συστήματός μας ή μπορούμε να επιλέξουμε την διαδρομή προς εγκατάσταση του πακέτου (other). Επίσης μπορούμε να αφαιρέσουμε πακέτα. Η επιλογή remove μας δείχνει όλα τα εγκατεστημένα πακέτα στο σύστημά μας.

4.5 Συστήματα ανίχνευσης εισβολέων (Intrusion Detection Systems, IDS)

Πολλά πακέτα προγραμμάτων έχουν αναπτυχθεί για την ανίχνευση διαφόρων ευπαθειών και εσφαλμένων ρυθμίσεων στα διάφορα λειτουργικά συστήματα. Τα προγράμματα αυτά ονομάζονται συστήματα ανίχνευσης εισβολέων (IDS). Σκοπός ενός συστήματος ανίχνευσης εισβολών είναι να εντοπίζει είτε την μη εξουσιοδοτημένη πρόσβαση ή την εσφαλμένη χρησιμοποίηση ενός πληροφοριακού συστήματος. Τα συστήματα αυτά είναι σημαντικό να αναφέρουμε ότι δεν υποκαθιστούν την λειτουργία του τείχους προστασίας (firewall). Άρα ένας γενικός ορισμός για τα συστήματα ανίχνευσης εισβολέων είναι ο εξής:

Είναι τα εργαλεία εκείνα τα οποία επιτρέπουν τον εντοπισμό των εισβολέων αλλά και των στοιχείων εκείνων που στο σύστημά μας δεν έχουν ρυθμιστεί σωστά ώστε να μας παρέχουν ασφάλεια και προστασία.

Ουσιαστικά τα εργαλεία αυτά ανάλογα με τον στόχο που έχουν (πχ. Δίκτυο, ακεραιότητα συστήματος κλπ) μας δίνουν διάφορες αναφορές σχετικά με την κατάσταση των διαφόρων συνιστώσεων του συστήματός μας.

Τα εργαλεία που θα χρησιμοποιήσουμε στο slackware είναι τα εξής:

1. Lynis
2. Chkrootkit
3. Rkhunter

4.5.1 Το εργαλείο Lynis

Το Lynis είναι ένα ανοιχτού κώδικα εργαλείο ασφαλείας. Είναι ένα ελεγκτικό εργαλείο συστήματος το οποίο μπορεί να τρέξει σε διάφορα λειτουργικά συστήματα (Linux, Mac OS X, BSD) και παρέχει καθοδήγηση με σκοπό την βελτίωση των ρυθμίσεων του συστήματος και την αποτροπή των εισβολών.
Το Lynis εκτελεί εκατοντάδες μεμονωμένες δοκιμές, για να καθορίσει την κατάσταση της ασφάλειας του συστήματος. Η ίδια η σάρωση ασφαλείας αποτελείται από την εκτέλεση μιας σειράς βημάτων, από την προετοιμασία του προγράμματος, μέχρι την εμφάνιση της έκθεσης/αναφοράς.

Τα βήματα που εκτελεί κατά την εκκίνηση του είναι τα εξής:

1. Καθορίζει το λειτουργικό σύστημα
2. Αναζητεί διαθέσιμα εργαλεία και βοηθητικά προγράμματα
3. Κάνει ελέγχους για ενημερώσεις του Lynis
4. Εκτελεί δοκιμές από ενεργοποιημένα plugins
5. Εκτελεί δοκιμές ασφαλείας ανά κατηγορία
6. Δημιουργεί την έκθεση για την κατάσταση της ασφάλειας του συστήματος

Κατά τη διάρκεια της σάρωσης, οι τεχνικές λεπτομέρειες σχετικά με τη σάρωση αποθηκεύονται σε ένα αρχείο καταγραφής. Την ίδια στιγμή οι διαπιστώσεις (προειδοποιήσεις, υποδείξεις, συλλογή δεδομένων), αποθηκεύονται σε ένα αρχείο αναφοράς.

Ένα σημαντικό στοιχείο που καθορίζει το συγκεκριμένο εργαλείο είναι η δυνατότητά του να ανιχνεύει και να χρησιμοποιεί εργαλεία τα οποία είναι εγκατεστημένα και έχουν μεγάλη σημασία από την οπτική της ασφάλειας. Για παράδειγμα αν εντοπίσει ότι τρέχουμε τον Apache στο σύστημά μας θα εκτελέσει και τους ανάλογους ελέγχους. Εάν κατά την διάρκεια ελέγχου του Apache εντοπιστούν ρυθμίσεις που αφορούν τα πρωτόκολλα SSL/TLS θα εκτελέσει ελέγχους και σε αυτά.

Δεδομένου ότι το Lynis είναι ευέλικτο, χρησιμοποιείται για αρκετούς διαφορετικούς σκοπούς. Κάποιες από τις τυπικές περιπτώσεις χρήσης του Lynis είναι οι εξής:

1. Έλεγχοι ασφαλείας
2. Δοκιμές συμμόρφωσης (π.χ. PCI, SOx)
3. Ανίχνευση ευπαθειών και σάρωση
4. Στεγάνωση του συστήματος

Το Linys είναι πιο αποτελεσματικό και περισσότερο σχολαστικό σε σχέση με το Nessus και άλλες παρόμοιες εφαρμογές. Λειτουργεί στον ίδιο τον host και ως εκ τούτου μπορεί να εκτελέσει μία βαθύτερη ανάλυση σε σύγκριση με σαρώσεις που έχουν ως βάση το δίκτυο. Επιπλέον δεν υπάρχει κίνδυνος για τις διάφορες διεργασίες που τρέχουν εκείνη την στιγμή στο σύστημα καθώς και για τα διάφορα log αρχεία διότι το lynis δεν προσπαθεί να συνδεθεί με αυτά.

Παρά το ότι το Lynis είναι ένα εργαλείο ελέγχου, μπορεί να ανακαλύψει τα τρωτά σημεία στο σύστημά μας που το κάνουν ευάλωτο στις διάφορες επιθέσεις. Αυτό επιτυγχάνεται με την χρήση υφιστάμενων εργαλείων και την ανάλυση των αρχείων ρυθμίσεων. Επιπλέον το Lynis είναι ένα ανοικτού κώδικα εργαλείο, σε αντίθεση με το Nessus το οποίο είναι κλειστού.

4.5.1.1 Εγκατάσταση και εκκίνηση

Το Lynis μπορούμε είτε να το εγκαταστήσουμε στο σύστημά μας είτε να το χρησιμοποιήσουμε χωρίς εγκατάσταση καθώς βασίζεται πάνω στο shell για να λειτουργήσει. Μπορούμε να κατεβάσουμε το συμπιεσμένο αρχείο από την επίσημη ιστοσελίδα:

https://cisofy.com/files/lynis-2.2.0.tar.gz

Έπειτα μπορούμε να αποσυμπιέσουμε το αρχείο στην τοποθεσία που επιθυμούμε. Στο παράδειγμά μας θα αποσυμπιέσουμε το αρχείο στην τοποθεσία `/home/diplomatiki/`.

Εάν θέλουμε να εγκαταστήσουμε το Lynis στο slackware μπορούμε να κατεβάσουμε το αρχείο από τα directories τις επίσημης ιστοσελίδας του slackware στην παρακάτω διεύθυνση:

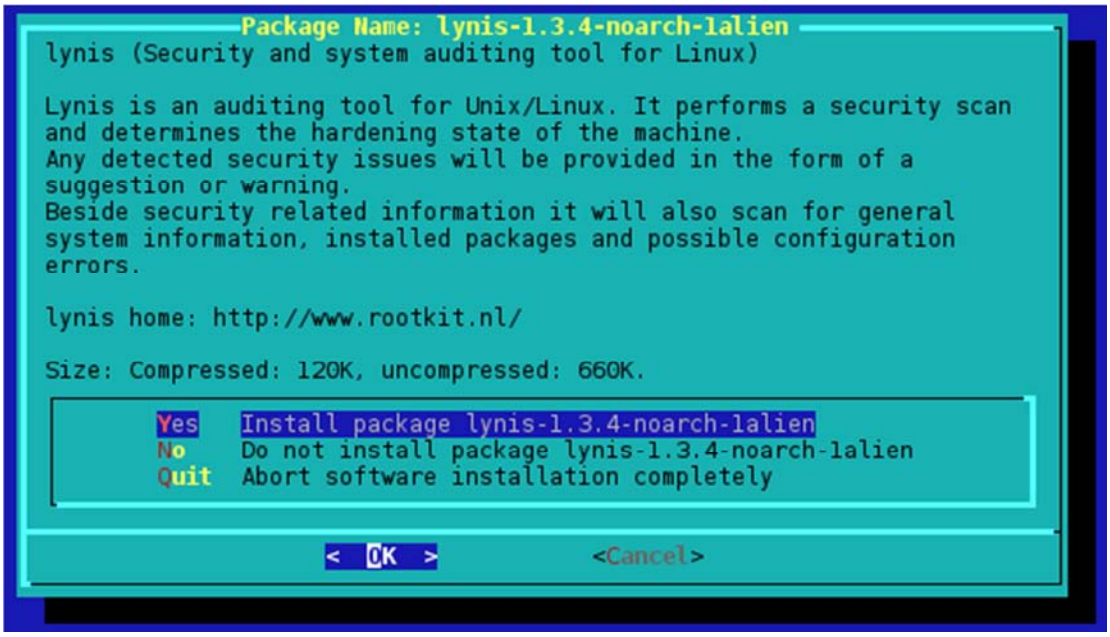
http://www.slackware.com/~alien/slackbuilds/lynis/pkg64/14.1/

επειδή χρησιμοποιούμε την έκδοση x64 του slackware θα κατεβάσουμε και την ανάλογη αρχιτεκτονική. Σε αυτό το σημείο θα πρέπει να αναφέρουμε ότι η έκδοση του lynis από την επίσημη ιστοσελίδα του slackware δεν βρίσκεται στη τελευταία της έκδοση και επίσης πώς την συγκεκριμένη τοποθεσία (directory) την διαχειρίζεται ένα από τα μέλη του slackware team ο Eric Hameleers τον οποίο σε πολλά forums τον βρίσκουμε με την ονομασία Alien Bob.

Η διαδικασία εγκατάστασης έχει ως εξής:

1. Κατεβάζουμε το αρχείο lynis-1.3.4-noarch-1alien.tgz
2. Ανοίγουμε το τερματικό μας
3. Αποκτούμε δικαιώματα διαχειριστή με την εντολή `su` και πληκτρολογούμε το password (Να σημειωθεί ότι κατά την πληκτρολόγηση δεν εμφανίζεται τίποτα)
4. Καλούμε το εργαλείο εγκατάστασης μέσω της εντολής `pkgtool` (ενότητα 4.4.2)
5. Επιλέγουμε current μιας και το αρχείο μας βρίσκεται στην διαδρομή /home/diplomatiki. Αν θέλουμε διαφορετική διαδρομή μπορούμε να την δώσουμε επιλέγοντας το other.
6. Και στη συνέχεια επιλέγουμε το ναι έτσι ώστε να γίνει η εγκατάσταση.

Στο σχήμα 4.20 μπορούμε να δούμε το παράθυρο εγκατάστασης μετά την επιλογή current.



Σχήμα 4.20 Παράθυρο εγκατάστασης του Lynis

Η εκκίνηση του εργαλείου γίνεται μέσω του τερματικού μας. Στην περίπτωση της εγκατάστασης μπορούμε να εκτελέσουμε το lynis μέσω της παρακάτω εντολής:

```
lynis –check-all
```

Με αυτή την εντολή το Lynis θα πραγματοποιήσει όλους τους διαθέσιμους ελέγχους στο σύστημά μας.
Αν χρειαστούμε βοήθεια σχετικά με τις παραμέτρους που μπορεί να δεχτεί το Lynis μπορούμε να το κάνουμε μέσω της εντολής:

```
Man lynis
```

Στην περίπτωση που θέλουμε να χρησιμοποιήσουμε το Lynis χωρίς εγκατάσταση (στην παρούσα διπλωματική) μπορούμε να εκτελέσουμε το εργαλείο με τις παρακάτω εντολές:

```
cd /home/diplomatiki/lynis
./lynis audit system
```

Η πρώτη εντολή μας οδηγεί στον κατάλογο που υπάρχουν τα αρχεία του Lynis ενώ η δεύτερη εκτελεί το εργαλείο.

4.5.1.2 Λειτουργία, ανίχνευση απειλών και διόρθωση ρυθμίσεων

Αφού δώσουμε τις παραπάνω εντολές το Lynis θα κάνει ανίχνευση των ρυθμίσεων του συστήματός μας και θα μας αποκαλύψει οτιδήποτε θα μπορούσαν επίδοξοι εισβολείς να χρησιμοποιήσουν εναντίων μας. Τα αρχικά στοιχεία που μας δίνει το Lynis είναι πληροφορίες για το σύστημά μας αλλά και τις διάφορες εκδόσεις που χρησιμοποιούμε καθώς και την έκδοση του Lynis που έχουμε και αν αυτή είναι ενημερωμένη ή όχι. Στο σχήμα 4.21 μπορούμε να δούμε τα αρχικά στοιχεία έτσι όπως αποτυπώθηκαν την πρώτη φορά που τρέξαμε το εργαλείο στο σύστημά μας.

Το εργαλείο μας ενημερώνει ότι η έκδοση του Lynis που χρησιμοποιούμε είναι η τελευταία δηλαδή η έκδοση 2.2.0 (Μάιος 2016). Μας δίνει πληροφορίες για το σύστημά μας όπως ποια έκδοση kernel χρησιμοποιούμε, ποια έκδοση linux, ποια αρχιτεκτονική, το hostname του συστήματος κλπ. Στην ενότητα boot and services δίνονται στοιχεία σχετικά με τις εκκινούμενες υπηρεσίες και διεργασίες. Όπως βλέπουμε η πρώτη διεργασία που χρησιμοποιεί ο πυρήνας (kernel) του συστήματός μας κατά την εκκίνησή του είναι η init. Δεν πραγματοποιείτε boot μέσω του UEFI καθώς ο υπολογιστής μας χρησιμοποιεί το BIOS, και ο εκκινητής ή το πρόγραμμα αρχικής φόρτωσης (ενότητα 1.3.1) του λειτουργικού μας συστήματος είναι το LILO.

```
[+] Initializing program
------------------------------------
  - Detecting OS...                                           [ DONE ]

  ----------------------------------------------------
  Program version:           2.2.0
  Operating system:          Linux
  Operating system name:     Slackware
  Operating system version:  14.1
  Kernel version:            3.10.17
  Hardware platform:         x86_64
  Hostname:                  diplomatiki
  Auditor:                   [Unknown]
  Profile:                   ./default.prf
  Log file:                  /var/log/lynis.log
  Report file:               /var/log/lynis-report.dat
  Report version:            1.0
  Plugin directory:          ./plugins
  ----------------------------------------------------
  - Checking profile file (./default.prf)...
  - Program update status...                                  [ NO UPDATE ]

[+] System Tools
------------------------------------
  - Scanning available tools...
  - Checking system binaries...

[+] Plugins (phase 1)
------------------------------------
 Note: plugins have more extensive tests, which may take a few minutes to complete

  - Plugins enabled                                           [ NONE ]

[+] Boot and services
------------------------------------
  - Service Manager                                           [ SysV Init ]
  - Checking UEFI boot                                        [ DISABLED ]
  - Checking presence LILO                                    [ OK ]
    - Password option presence                                [ WARNING ]
  - Check startup files (permissions)                         [ OK ]
```

Σχήμα 4.21 Αρχικά στοιχεία σάρωσης του συστήματος με το Lynis

Το Lynis επίσης μας προειδοποιεί ότι κατά την εκκίνηση είναι σημαντικό να χρησιμοποιούμε κάποιο password έτσι ώστε να μην είναι δυνατόν:

1. Επίδοξοι εισβολείς να μπορούν να χρησιμοποιήσουν ένα Live CD για να αποκτήσουν πρόσβαση στις διάφορες κατατμήσεις του Linux.
2. Να εκκινήθει το σύστημά μας σε "κατάσταση ενός χρήστη" (runlevel 1 που σημαίνει ότι κάθε χρήστης μπορεί να δώσει εντολές ως διαχειριστής χωρίς να χρειάζεται κωδικό πρόσβασης)
3. Να εκκινείτε το LILO χωρίς password.

Για να μπορέσουμε να διορθώσουμε και να ρυθμίσουμε το password αρκεί να επεξεργαστούμε το αρχείο που υπάρχει στην διαδρομή /etc/lilo.conf με τον επεξεργαστεί nano όπως έχουμε κάνει και σε άλλες περιπτώσεις. Άρα η εντολή που θα χρησιμοποιήσουμε είναι η εξής:

```
nano /etc/lilo.conf
```

Έπειτα θα εντοπίσουμε το παρακάτω σημείο κώδικα και κάτω από την εντολή `image=/boot/vmlinuz` θα προσθέσουμε την εντολή `password=`κωδικός που θέλουμε να έχουμε. Στην περίπτωσή μας θα χρησιμοποιήσουμε τον κωδικό: diploma123!

```
# Linux bootable partition config begins
image = /boot/vmlinuz
password=diploma123!
root = /dev/sda2
label = Linux
read-only
# Linux bootable partition config ends
```

Στη συνέχεια θα αποθηκεύσουμε το αρχείο αυτό πατώντας το πλήκτρο F3 και στην συνέχεια θα επιβεβαιώσουμε την αποθήκευση. Έπειτα θα πατήσουμε το πλήκτρο F2 για να εξέλθουμε από το nano. Για να μπορέσει να καταγραφεί και να ανανεωθεί το αρχείο αυτό στο MBR (master boot record-τομέας εκκίνησης) θα πρέπει επίσης να εκτελέσουμε την παρακάτω εντολή:

```
/sbin/lilo
```

Κατά την επόμενη εκκίνηση θα μας ζητηθεί κωδικός πρόσβασης προκειμένου να εκκινηθεί το λειτουργικό μας σύστημα. Αν τρέξουμε και πάλι το Lynis μετά την ρύθμιση του παραπάνω κωδικού δεν θα μας δώσει καμία προειδοποίηση σχετικά με το password για τον εκκινητή LILO. Σε αυτό το σημείο θα πρέπει να αναφέρουμε πώς διάφορες υποδείξεις για το πώς να διορθώσουμε τα διάφορα στοιχεία που μας έχει εμφανίσει το εργαλείο μπορούμε να βρούμε στο τέλος του ελέγχου που πραγματοποιήσαμε με το Lynis.

Στο σχήμα 4.22 βλέπουμε την συνέχεια από τις σαρώσεις που πραγματοποιεί το Lynis. Όπως μπορούμε να παρατηρήσουμε ο πυρήνας μας και το λειτουργικό μας σύστημα τρέχει στο επίπεδο 3. Το επίπεδο που τρέχει το κάθε λειτουργικό σύστημα μπορεί να διαφέρει, στο slackware έχουμε τα παρακάτω επίπεδα:

ID	Περιγραφή
0	Συντήρηση
1	Λειτουργία Single-user
2	Αχρησιμοποίητο αλλά είναι το ίδιο με το επίπεδο 3
3	Λειτουργία Multi-user χωρίς γραφικό περιβάλλον
4	Λειτουργία Multi-user με γραφικό περιβάλλον (X11)
5	Αχρησιμοποίητο αλλά είναι το ίδιο με το επίπεδο 3
6	Reboot

Πίνακας 4.1 Επίπεδα (run level) στο slackware

```
[+] Kernel
------------------------------------
  - Checking default run level                                [ 3 ]
  - Checking CPU support (NX/PAE)
    CPU support: PAE and/or NoeXecute supported               [ FOUND ]
  - Checking kernel version and release                       [ DONE ]
  - Checking kernel type                                      [ DONE ]
  - Checking loaded kernel modules                            [ DONE ]
      Found 44 active modules
  - Checking Linux kernel configuration file                  [ FOUND ]
  - Check if reboot is needed                                 [ NO ]

[ Press [ENTER] to continue, or [CTRL]+C to stop ]


[+] Memory and processes
------------------------------------
  - Checking /proc/meminfo                                    [ FOUND ]
  - Searching for dead/zombie processes                       [ OK ]
  - Searching for IO waiting processes                        [ OK ]

[ Press [ENTER] to continue, or [CTRL]+C to stop ]


[+] Users, Groups and Authentication
------------------------------------
  - Search administrator accounts                             [ OK ]
  - Checking for non-unique UIDs                              [ OK ]
  - Checking consistency of group files (grpck)               [ WARNING ]
  - Checking non unique group ID's                            [ OK ]
  - Checking non unique group names                           [ OK ]
  - Checking password file consistency                        [ OK ]
  - Query system users (non daemons)                          [ DONE ]
  - Checking NIS+ authentication support                      [ NOT ENABLED ]
  - Checking NIS authentication support                       [ NOT ENABLED ]
  - Checking sudoers file                                     [ FOUND ]
    - Check sudoers file permissions                          [ OK ]
  - Checking PAM password strength tools                      [ SUGGESTION ]
  - Checking PAM configuration file (pam.conf)                [ NOT FOUND ]
  - Checking PAM configuration files (pam.d)                  [ NOT FOUND ]
  - Checking PAM modules                                      [ NOT FOUND ]
  - Checking accounts without expire date                     [ OK ]
  - Checking accounts without password                        [ OK ]
  - Checking user password aging (minimum)                    [ DISABLED ]
  - Checking user password aging (maximum)                    [ DISABLED ]
  - Checking expired passwords                                [ FOUND ]
```

Σχήμα 4.22 Δεύτερη οθόνη σάρωσης με το Lynis

Μας ενημερώνει επίσης για την Physical Address Extension (PAE), για το αν είναι ενημερωμένος ο kernel και αν είναι στην τελευταία έκδοση τις σειράς του κλπ. Στην ενότητα Memory and processes μπορούμε να δούμε αν οι λειτουργίες τις μνήμης και των διεργασιών είναι σωστές, αν υπάρχουν διεργασίες Zombie που εκτελούνται κλπ (κεφάλαιο 2). Η ενότητα users, groups and authentication μας δίνει πληροφορίες σχετικά με τους χρήστες, τα group που είναι ενταγμένοι οι διάφοροι χρήστες και τη βελτιστοποίηση των ρυθμίσεων αυτών. Βλέπουμε πώς έχουμε μία προειδοποίηση σχετικά με τα αρχεία των group που έχουν δημιουργηθεί, η οποία ουσιαστικά επαληθεύει την ακεραιότητα των πληροφοριών των διαφόρων ομάδων και κωδικών πρόσβασης. Ελέγχει ότι όλες οι εγγραφές στο /etc/group και /etc/gshadow έχουν την κατάλληλη μορφή και περιέχουν έγκυρα δεδομένα καθώς και

ότι οι κωδικοί πρόσβασής μας είναι κρυπτογραφημένοι. Η επίλυση της συγκεκριμένης προειδοποίησης μπορεί να γίνει με το παρακάτω αυτόματο τρόπο:

Ανοίγουμε το τερματικό με δικαιώματα διαχειριστή. Εκτελούμε την εντολή:

```
grpconv
```

Η εντολή αυτή θα δημιουργήσει ένα κρυφό αρχείο ομάδων έτσι ώστε να κρατήσει κρυπτογραφημένους τους κωδικούς πρόσβασής μας. Επίσης με αυτόν τον τρόπο οι κωδικοί μας θα είναι ασφαλείς από προγράμματα που προσπαθούν να τους σπάσουν (password cracking programs). Η εντολή `grpconv` δημιουργεί το αρχείο /etc/gshadow το οποίο βασίζεται σε ένα ήδη υπάρχον αρχείο το /etc/groups και αντικαθιστά τις κρυπτογραφημένες καταχωρήσεις των κωδικών πρόσβασής μας με τους χαρακτήρες x. Αν προσθέσουμε νέες καταχωρήσεις στο αρχείο /etc/groups, μπορούμε να εκτελέσουμε και πάλι την εντολή `grpconv` για να μεταφέρει τις νέες πληροφορίες στο /etc/gshadow. Τις επόμενες φορές που εκτελέσουμε την εντολή αυτή οι καταχωρήσεις που έχουν ήδη έναν κωδικό πρόσβασης (x) θα αγνοηθούν και θα γίνει η μετατροπή μόνο εκείνων που δεν έχουν. Εάν επιθυμούμαι την επαναφορά των κωδικών μας μπορούμε να εκτελέσουμε την εντολή `grpconv` η οποία θα αποκαταστήσει τους κρυπτογραφημένους κωδικούς πρόσβασης στο αρχείο /etc/groups και θα αφαιρέσει το αρχείο /etc/gshadow.

Στη συνέχεια βλέπουμε ότι το σύστημά μας δεν χρησιμοποιεί την NIS και NIS+ υπηρεσία καταλόγου. Οι ρυθμίσεις για τους διαχειριστές του συστήματος και τα δικαιώματα που τους έχουν εκχωρηθεί είναι σωστά. Έχουμε μία προτεινόμενη ενέργεια σχετικά με τις βιβλιοθήκες PAM (Pluggable Authentication Module). Πρόκειται ουσιαστικά για ένα framework που χρησιμοποιείται για την πιστοποίηση των χρηστών σε διάφορες εφαρμογές. Το pam_cracklib είναι ένα μέρος του PAM και σύμφωνα με την εντολή `man`, "μπορεί να συνδεθεί στη στοίβα του κωδικού μιας δεδομένης εφαρμογής έτσι ώστε να παρέχει έλεγχο για τους κωδικούς πρόσβασης". Στην συγκεκριμένη έκδοση του slackware ένα τέτοιο module δεν υπάρχει. Στο τέλος της σάρωσης του Lynis μας ενημερώνει και μας προτείνει να εγκαταστήσουμε ένα τέτοιο module. Το Lynis μας προτείνει το pam_cracklib.

Οι χρήστες τείνουν να βασίζουν τους κωδικούς πρόσβασής τους σε καθημερινές λέξεις και φράσεις. Το CrackLib είναι ένα εργαλείο το οποίο μπορεί να φιλτράρει τις λέξεις και φράσεις που είναι γνώριμες στους cracker, χρησιμοποιώντας ένα λεξικό που δημιουργήθηκε από λίστες λέξεων και φράσεων. Υπάρχουν πολλές προεπιλεγμένες λίστες λέξεων και φράσεων στο internet για λήψη κυρίως όμως οι λέξεις και φράσεις αυτές είναι στα αγγλικά. Σε πολλές περιπτώσεις, μπορεί να είναι απαραίτητο να κατεβάσουμε (ή ακόμη και να δημιουργήσουμε) επιπλέον λίστες λέξεων στα Ελληνικά.

Στην παρακάτω διεύθυνση μπορούμε να βρούμε περισσότερες πληροφορίες σχετικά με το cracklib το οποίο χρήζει μεγαλύτερης και εκτενέστερης έρευνας.

ftp://ftp.slackware.com/pub/lfs-website/blfs/view/svn/postlfs/cracklib.html

Στη συνέχεια το εργαλείο μας ενημερώνει ότι υπάρχουν λογαριασμοί που οι κωδικοί πρόσβασης έχουν λήξη. Στο τέλος της σάρωσης βλέπουμε πώς αυτός είναι ο root χρήστης τον οποίο για λόγους ασφαλείας θα πρέπει να αλλάζουμε το password του συχνά. Μπορούμε να το κάνουμε αυτό δίνοντας τις παρακάτω εντολές:

```
su
passwd
```

στη συνέχεια θα μας ζητηθεί να ορίσουμε το νέο μας κωδικό για τον χρήστη root. Θα πρέπει να προσέξουμε ιδιαίτερα καθώς εφόσον πληκτρολογήσουμε την εντολή `passwd` θα πρέπει οπωσδήποτε να ορίσουμε κάποιον κωδικό αλλιώς θα μείνει κενός. Στην ενότητα shells (σχήμα 4.23) το εργαλείο μας προειδοποιεί για διάφορες ευπάθειες που προκύπτουν από μη ενημερωμένα πακέτα των shells. Οι συγκεκριμένες ευπάθειες αφορούν αποκλειστικά τα κελύφη που υπάρχουν εγκατεστημένα στο σύστημά μας και τις εξαρτήσεις αυτών. Η διόρθωση είναι απλή και γίνεται με την ενημέρωση του συστήματός μας (ενότητα 4.4) για να έχουμε τις τελευταίες ενημερώσεις των διαφόρων προγραμμάτων, στοιχείων και εφαρμογών που καλύπτουν τα κενά που έχουν βρεθεί. Για να μάθουμε περισσότερα για τις ευπάθειες αυτές μπορούμε να επισκεφτούμε τον παρακάτω ιστότοπο:

https://cve.mitre.org/

κάνοντας αναζήτηση με τον κωδικό που μας παρουσιάζεται στην σάρωση του Lynis μπορούμε να δούμε τι ακριβώς αφορούν οι συγκεκριμένες ευπάθειες. Επίσης όπως αναφέρθηκε και προηγουμένως στο τέλος της ανάλυσης του Lynis υπάρχουν προτεινόμενες λύσεις και πηγές από όπου μπορούμε να πάρουμε τις απαραίτητες πληροφορίες.

Στις ενότητες File system και Storage γίνονται αναφορές σχετικά με τα αρχεία του συστήματος και τις συνδεόμενες συσκευές. Το Lynis μας προτείνει να μην γίνεται αυτόματη προσάρτηση των αφαιρούμενων μονάδων έτσι ώστε να μπορούμε να κάνουμε σάρωση των μονάδων αυτών πριν εκτελεστούν από το σύστημα, καθώς μπορεί να περιέχουν αρχεία τα οποία είναι κακόβουλα.

```
[+] Shells
------------------------------------
  - Checking shells from /etc/shells
    Result: found 6 shells (valid shells: 6).
    - Session timeout settings/tools                          [ NONE ]
  - Checking default umask values
    - Checking default umask in /etc/profile                  [ WEAK ]
  - Shellshock: CVE-2014-6271 (original shellshocker)         [ WARNING ]
  - Shellshock: CVE-2014-6278 (Florian's patch, lcamtuf bug #2)2C [ WARNING ]
  - Shellshock: CVE-2014-7186 redir_stack bug                 [ WARNING ]

[ Press [ENTER] to continue, or [CTRL]+C to stop ]


[+] File systems
------------------------------------
  - Checking mount points
    - Checking /home mount point                              [ OK ]
    - Checking /tmp mount point                               [ SUGGESTION ]
    - Checking /var mount point                               [ SUGGESTION ]
  - Checking LVM volume groups                                [ NONE ]
  - Querying FFS/UFS mount points (fstab)                     [ NONE ]
  - Query swap partitions (fstab)                             [ OK ]
  - Testing swap partitions                                   [ OK ]
  - Checking for old files in /tmp                            [ OK ]
  - Checking /tmp sticky bit                                  [ OK ]
  - ACL support root file system                              [ ENABLED ]
  - Checking Locate database                                  [ FOUND ]

[ Press [ENTER] to continue, or [CTRL]+C to stop ]


[+] Storage
------------------------------------
  - Checking usb-storage driver (modprobe config)             [ NOT DISABLED ]
  - Checking USB devices authorization                        [ ENABLED ]
  - Checking firewire ohci driver (modprobe config)           [ NOT DISABLED ]

[ Press [ENTER] to continue, or [CTRL]+C to stop ]


[+] NFS
------------------------------------
  - Check running NFS daemon                                  [ NOT FOUND ]
```

Σχήμα 4.23 Τρίτη οθόνη σάρωσης με το Lynis

Στο σχήμα 4.24 η ενότητα Name services περιέχει τις υπηρεσίες δικτύου που χρησιμοποιούμαι. Μπορούμε να δούμε πώς δεν έχουμε ενεργοποιημένες υπηρεσίες στο σύστημά μας τύπου Nscd ο οποίος είναι ένας δαίμονας της μνήμης cash. Το αρχείο hosts (χρησιμοποιείται για να αντιστοιχίζονται ονόματα υπολογιστών σε διευθύνσεις IP) δεν περιέχει διπλότυπες εγγραφές. Η ενότητα ports and packages μας ενημερώνει ότι στο σύστημά μας υπάρχει εγκατεστημένος ένας διαχειριστής πακέτων RPM ο οποίος μπορεί να διαχειριστεί, να εγκαταστήσει και να απεγκαταστήσει αρχεία με κατάληξη .rpm. Εκτός από το Lynis το εργαλείο δεν μπορεί να εντοπίσει κάποια άλλη εφαρμογή audit tool στο σύστημά μας. Στην ενότητα Networking το πρωτόκολλο IPv6 είναι ενεργοποιημένο και είναι στην αυτόματη λειτουργία, επίσης μπορούμε να δούμε πώς 9 πόρτες είναι ανοικτές. Μία καλή πρακτική για να δούμε τις πόρτες που είναι ανοικτές στο slackware είναι η παρακάτω:

```
[+] Name services
------------------------------------
  - Checking default DNS search domain                    [ NONE ]
  - Checking /etc/resolv.conf options                     [ NONE ]
  - Searching DNS domain name                             [ FOUND ]
      Domain name:diplomatiki
  - Checking nscd status                                  [ NOT FOUND ]
  - Checking Unbound status                               [ NOT FOUND ]
  - Checking BIND status                                  [ NOT FOUND ]
  - Checking PowerDNS status                              [ NOT FOUND ]
  - Checking ypbind status                                [ NOT FOUND ]
  - Checking /etc/hosts
    - Checking /etc/hosts (duplicates)                    [ OK ]
    - Checking /etc/hosts (hostname)                      [ OK ]
    - Checking /etc/hosts (localhost)                     [ OK ]

[ Press [ENTER] to continue, or [CTRL]+C to stop ]


[+] Ports and packages
------------------------------------
  - Searching package managers
    - Searching RPM package manager                       [ FOUND ]
      - Querying RPM package manager
  - Checking package audit tool                           [ NONE ]

[ Press [ENTER] to continue, or [CTRL]+C to stop ]


[+] Networking
------------------------------------

  - Checking IPv6 configuration                           [ ENABLED ]
      Configuration method                                [ AUTO ]
      IPv6 only                                           [ NO ]
  - Checking configured nameservers
    - Testing nameservers
        Nameserver: 192.168.1.1                           [ OK ]
        Nameserver: 10.0.5.3                              [ OK ]
    - Minimal of 2 responsive nameservers                 [ OK ]
  - Checking default gateway                              [ DONE ]
  - Getting listening ports (TCP/UDP)                     [ DONE ]
      * Found 9 ports
  - Checking promiscuous interfaces                       [ OK ]
  - Checking waiting connections                          [ OK ]
  - Checking status DHCP client                           [ NOT ACTIVE ]
  - Checking for ARP monitoring software                  [ NOT FOUND ]
```

Σχήμα 4.24 Τέταρτη οθόνη σάρωσης με το Linys

Ανοίγουμε το τερματικό και αποκτάμε δικαιώματα διαχειριστή. Στη συνέχεια καλούμε το nmap το οποίο έρχεται προεγκατεστημένο με το σύστημά μας.

```
Nmap localhost
```

Η παραπάνω εντολή θα μας δείξει τις εσωτερικές ip οι οποίες είναι ανοικτές στο εσωτερικό μας σύστημα (σχήμα 4.25). Για να δούμε τις πόρτες οι οποίες είναι ανοικτές και κλειστές στο επίπεδο του δικτύου, θα πρέπει πρώτα να γνωρίζουμε την

εξωτερική διεύθυνση ip του modem/router. Για να το δούμε αυτό αρκεί να επιστεφτούμε μία ιστοσελίδα που μας πληροφορεί για την εξωτερική μας διεύθυνση. Έτσι με την παρακάτω εντολή στο τερματικό μας μπορούμε να δούμε τις εσωτερικές διευθύνσεις στο επίπεδο δικτύου (σχήμα 4.26).

```
nmap 85.75.156.122
```

```
bash-4.2# nmap localhost

Starting Nmap 6.40 ( http://nmap.org ) at 2016-05-27 07:54 EEST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000013s latency).
Not shown: 995 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
37/tcp   open  time
113/tcp  open  ident
6000/tcp open  X11
6881/tcp open  bittorrent-tracker

Nmap done: 1 IP address (1 host up) scanned in 0.13 seconds
```

Σχήμα 4.25 Αποτελέσματα τις εντολής nmap localhost

```
bash-4.2# nmap 85.75.156.122

Starting Nmap 6.40 ( http://nmap.org ) at 2016-05-27 07:56 EEST
Nmap scan report for athedsl-156892.home.otenet.gr (85.75.156.122)
Host is up (0.0028s latency).
Not shown: 993 filtered ports
PORT     STATE  SERVICE
21/tcp   open   ftp
23/tcp   closed telnet
53/tcp   open   domain
80/tcp   open   http
161/tcp  closed snmp
445/tcp  closed microsoft-ds
5555/tcp open   freeciv

Nmap done: 1 IP address (1 host up) scanned in 4.86 seconds
```

Σχήμα 4.26 Αποτελέσματα τις εντολής nmap ip

Όπως μπορούμε να δούμε (σχήμα 4.25) η πόρτα 22 του tcp χρησιμοποιείται από το ssh το οποίο είναι ένα (ασφαλές) δικτυακό πρωτόκολλο το οποίο επιτρέπει τη μεταφορά δεδομένων μεταξύ δύο υπολογιστών. Όπως θα πούμε και σε επόμενη ενότητα το πρωτόκολλο αυτό δέχεται πολλές επιθέσεις και καλό θα είναι να αλλάξουμε τις ρυθμίσεις και την πόρτα στην οποία ακούει. Σε επόμενη ενότητα γίνεται ανάλυση αυτής της μεθόδου.

Βλέπουμε επίσης ότι η πόρτα 6881 είναι ανοικτή για τις υπηρεσίες που χρησιμοποιεί το πρόγραμμα torrent. Καλό θα είναι σε περιπτώσεις server η πόρτα αυτή να είναι κλειστή. Σε επόμενη ενότητα θα δούμε πώς μπορούμε να κλείσουμε μία πόρτα στο σύστημά μας.

Στο σχήμα 4.26 η πόρτα 22 του tcp χρησιμοποιείται από τον ftp. Η πόρτα που χρησιμοποιείτε από την ξεπερασμένη υπηρεσία telnet όπως βλέπουμε είναι κλειστή

εκ των προτέρων στο σύστημα του slackware. Καλό είναι βέβαια να απενεργοποιούμε την υπηρεσία εντελώς. Σε επόμενη ενότητα θα δούμε πώς μπορούμε να το κάνουμε αυτό.

Όπως επίσης μπορούμε να δούμε ανοικτή είναι και η πόρτα 5555. Η πόρτα αυτή στην περίπτωσή μας χρησιμοποιείτε από το nmap που υπάρχει εγκατεστημένο, όμως πολλές φορές η πόρτα αυτή χρησιμοποιείται από επίδοξους εισβολείς για να εισχωρήσουν στο σύστημά μας. Χαρακτηριστικό παράδειγμα αποτελεί ο δούρειος ίππος (Trojan horse) Backdoor.Darkmoon.E που ανακαλύφθηκε στις 25-09-2007, όπου ανοίγει μια κερκόπορτα στη θύρα TCP 5555 για την παραβίαση του υπολογιστή. Σύμφωνα βέβαια με την Symantec δεν επηρεάζει τα συστήματα του Linux και του Unix παρά μόνο των Windows. Βέβαια αυτό δεν σημαίνει πώς δεν μπορεί να υπάρξει παραλλαγή του και να επηρεάσει περισσότερα συστήματα. Πολλές άλλες περιπτώσεις εισβολέων που χρησιμοποιούν την συγκεκριμένη πόρτα υπάρχουν, όμως δεν θα επεκταθούμε περαιτέρω.

```
[+] Printers and Spools
------------------------------------
  - Checking cups daemon                                    [ NOT FOUND ]
  - Checking lp daemon                                      [ NOT RUNNING

[ Press [ENTER] to continue, or [CTRL]+C to stop ]


[+] Software: e-mail and messaging
------------------------------------
  - Checking Exim status                                    [ NOT FOUND ]
  - Checking Postfix status                                 [ NOT FOUND ]
  - Checking Dovecot status                                 [ NOT FOUND ]
  - Checking Qmail status                                   [ NOT FOUND ]
  - Checking Sendmail status                                [ NOT FOUND ]

[ Press [ENTER] to continue, or [CTRL]+C to stop ]


[+] Software: firewalls
------------------------------------
  - Checking iptables kernel module                         [ NOT FOUND ]
  - Checking host based firewall                            [ NOT ACTIVE ]

[ Press [ENTER] to continue, or [CTRL]+C to stop ]


[+] Software: webserver
------------------------------------
  - Checking Apache (binary /usr/sbin/httpd)                [ FOUND ]
      Info: Configuration file found (/etc/httpd/httpd.conf)
      Info: Found 8 virtual hosts
    * Loadable modules                                      [ FOUND ]
        - Found 109 loadable modules
          mod_evasive: anti-DoS/brute force                 [ NOT FOUND ]
          mod_qos: anti-Slowloris                           [ NOT FOUND ]
          mod_spamhaus: anti-spam (spamhaus)                [ NOT FOUND ]
          ModSecurity: web application firewall             [ NOT FOUND ]
  - Checking nginx                                          [ NOT FOUND ]
```

Σχήμα 4.27 Πέμπτη οθόνη σάρωσης με το Lynis

Σημαντικές ενότητες αποτελούν τα firewalls και webserver software. Στην πρώτη περίπτωση το Lynis μας προειδοποιεί για την μη ύπαρξη κάποιου ενεργού τείχους προστασίας και ότι ο iptables δεν είναι ενεργός. Σε επόμενη ενότητα θα δούμε πώς ενεργοποιούμε το τείχος προστασίας στο slackware καθώς επίσης θα προσπαθήσουμε να θέσουμε κανόνες και ένα πλαίσιο ασφαλείας στο τείχος μέσω του iptables.

Στην δεύτερη ενότητα ο apache είναι ενεργός στο σύστημά μας όμως δεν έχουμε λάβει καθόλου μέτρα προστασίας σε περιπτώσεις επιθέσεων προς αυτόν. Τα mod_evasive και ModSecurity είναι δωρεάν modules του Apache τα οποία προστατεύουν τον web server από διάφορες επιθέσεις ωμής βίας (αναφέρεται στην εξαντλητική δοκιμή πιθανών κλειδιών που παράγουν ένα κρυπτογράφημα) ή επιθέσεις DDoS (οι επιθέσεις εναντίον ενός υπολογιστή, ή μιας υπηρεσίας που παρέχεται, οι οποίες έχουν ως σκοπό να καταστήσουν τον υπολογιστή ή την υπηρεσία ανίκανη να δεχτεί άλλες συνδέσεις και έτσι να μην μπορεί να εξυπηρετήσει άλλους πιθανούς πελάτες), συμπεριλαμβανομένων των SQL injection (η δυνατότητα σε κάποιον κακόβουλο επιτιθέμενο να “τρέξει” εντολές SQL ενάντια σε ένα server – στόχο και στη συνέχεια να αποσπάσει αρκετά ευαίσθητες πληροφορίες (όπως για παράδειγμα κωδικοί πρόσβασης, ονόματα χρηστών, emails, αριθμοί πιστωτικών καρτών κ.α) μέσα από την βάση δεδομένων στην οποία και επιτίθεται), cross-site scripting (αναφέρεται στην εκμετάλλευση διάφορων ευπαθειών (vulnerabilities) υπολογιστικών συστημάτων με εισαγωγή κώδικα HTML ή Javascript σε κάποιο ιστοχώρο), κα. Μπορούμε να κατεβάσουμε τα εν λόγω εργαλεία από διάφορες πηγές των κοινοτήτων του slackware και να κάνουμε την εγκατάστασή τους με τον τρόπο που υποδείξαμε στην ενότητα 4.4.2.

Όπως και τα παραπάνω modules έτσι και το mod_qos ενεργεί σε περιπτώσεις DDoS επιθέσεων. Το συγκεκριμένο εργαλείο ενεργεί στην περίπτωση επίθεσης Slowloris η οποία είναι επίθεση τύπου DDoS που εφευρέθηκε από τον Robert "RSnake" Hansen, και επιτρέπει σε ένα μηχάνημα να τερματίσει τις παρεχόμενες υπηρεσίες ενός άλλου μηχανήματος web server με ελάχιστο εύρος ζώνης.

Η ενότητα SSH support (σχήμα 4.28) περιέχει τις ρυθμίσεις που έχουν γίνει στο πρωτόκολλο SSH (Secure shell) καθώς και το αν είναι ενεργό το module που παρέχει την συγκεκριμένη υπηρεσία. Μπορούμε να δούμε τις ρυθμίσεις αυτές στην διαδρομή `/etc/ssh/sshd_conf.` Αναλυτικότερα για τις διάφορες ρυθμίσεις στην ενότητα 4.6.

Το Simple Network Management Protocol (SNMP) στην ενότητα SNMP support είναι μέρος της σουίτας πρωτοκόλλων Internet (IP - Internet Protocol). Χρησιμοποιείται στα συστήματα διαχείρισης δικτύων, στη διαχείριση και παρακολούθηση δικτυακών συσκευών που απαιτούν παρέμβαση του διαχειριστή δικτύου. Αποτελείται από μια ομάδα προτύπων για τη διαχείριση δικτύου και περιλαμβάνει ένα πρωτόκολλο επιπέδου εφαρμογών (application layer), ένα σχήμα βάσης δεδομένων και μια ομάδα από σύνολα δεδομένων. Στο σύστημά μας δεν τρέχει ο συγκεκριμένος δαίμονας. Μπορούμε να δούμε τις ρυθμίσεις αυτής τις υπηρεσίας στην διαδρομή `/etc/net-snmp/snmpd.conf.`

Η ενότητα Databases μας ενημερώνει για τις εγκατεστημένες βάσεις δεδομένων. Όπως βλέπουμε η βάση δεδομένων MySQL είναι προστατευμένη με κωδικό για αποτροπή περιπτώσεων μη εξουσιοδοτημένης χρήσης. Οι βάσεις δεδομένων PostgreSQL και Oracle δεν είναι εγκατεστημένες στο σύστημά μας.

```
[+] SSH Support
------------------------------------
  - Checking running SSH daemon                              [ FOUND ]
    - Searching SSH configuration                            [ FOUND ]
    - SSH option: AllowTcpForwarding                         [ NOT FOUND ]
    - SSH option: ClientAliveCountMax                        [ NOT FOUND ]
    - SSH option: ClientAliveInterval                        [ NOT FOUND ]
    - SSH option: Compression                                [ NOT FOUND ]
    - SSH option: FingerprintHash                            [ NOT FOUND ]
    - SSH option: GatewayPorts                               [ NOT FOUND ]
    - SSH option: IgnoreRhosts                               [ NOT FOUND ]
    - SSH option: LoginGraceTime                             [ NOT FOUND ]
    - SSH option: LogLevel                                   [ NOT FOUND ]
    - SSH option: MaxAuthTries                               [ NOT FOUND ]
    - SSH option: MaxStartups                                [ NOT FOUND ]
    - SSH option: MaxSessions                                [ NOT FOUND ]
    - SSH option: PermitRootLogin                            [ NOT FOUND ]
    - SSH option: PermitUserEnvironment                      [ NOT FOUND ]
    - SSH option: PermitTunnel                               [ NOT FOUND ]
    - SSH option: Port                                       [ NOT FOUND ]
    - SSH option: PrintLastLog                               [ NOT FOUND ]
    - SSH option: Protocol                                   [ NOT FOUND ]
    - SSH option: StrictModes                                [ NOT FOUND ]
    - SSH option: TCPKeepAlive                               [ NOT FOUND ]
    - SSH option: UseDNS                                     [ NOT FOUND ]
    - SSH option: UsePrivilegeSeparation                     [ OK ]
    - SSH option: VerifyReverseMapping                       [ NOT FOUND ]
    - SSH option: X11Forwarding                              [ NOT FOUND ]
    - SSH option: AllowUsers                                 [ NOT FOUND ]
    - SSH option: AllowGroups                                [ NOT FOUND ]

[ Press [ENTER] to continue, or [CTRL]+C to stop ]


[+] SNMP Support
------------------------------------
  - Checking running SNMP daemon                             [ NOT FOUND ]

[ Press [ENTER] to continue, or [CTRL]+C to stop ]


[+] Databases
------------------------------------
  - MySQL process status                                     [ FOUND ]
    - Checking MySQL root password                           [ OK ]
  - PostgreSQL processes status                              [ NOT FOUND ]
  - Oracle processes status                                  [ NOT FOUND ]
```

Σχήμα 4.28 Έκτη οθόνη σάρωσης με το Lynis

Για λόγους συντομίας στο σχήμα 4.29 συμπεριλήφθησαν οι σημαντικότερες ενότητες του Lynis καθώς όπως αναφέραμε και στα αρχικά σχόλια το Lynis εκτελεί εκατοντάδες ελέγχους και σαρώσεις.

Η ενότητα Logging and files μας πληροφορεί ότι ο δαίμονας Log είναι ενεργοποιημένος στο σύστημά μας. Ο Daemon Log είναι ένα χρήσιμο εργαλείο, διότι μπορεί να διαγνώσει πολλά λάθη διαμόρφωσης, ρυθμίσεων και θέματα λογισμικού.

Ο klogd διαβάζει τα μηνύματα καταγραφής του πυρήνα και βοηθά στην διαδικασία να αποστείλει τα μηνύματα αυτά στα κατάλληλα αρχεία ή τους χρήστες.

Η υπηρεσία απομακρυσμένης διαχείρισης SSH remote δεν είναι ενεργοποιημένη στο σύστημα.

```
[+] Logging and files
------------------------------------
  - Checking for a running log daemon                        [ OK ]
    - Checking Syslog-NG status                              [ NOT FOUND ]
    - Checking systemd journal status                        [ NOT FOUND ]
    - Checking Metalog status                                [ NOT FOUND ]
    - Checking RSyslog status                                [ NOT FOUND ]
    - Checking RFC 3195 daemon status                        [ NOT FOUND ]
    - Checking klogd                                         [ FOUND ]
    - Checking minilogd instances                            [ NOT FOUND ]
  - Checking logrotate presence                              [ OK ]
  - Checking remote logging                                  [ NOT ENABLED ]
  - Checking log directories (static list)                   [ DONE ]
  - Checking open log files                                  [ DONE ]
  - Checking deleted files in use                            [ FILES FOUND ]

[ Press [ENTER] to continue, or [CTRL]+C to stop ]


[+] Insecure services
------------------------------------
  - Checking inetd status                                    [ ACTIVE ]
    - Checking inetd.conf                                    [ FOUND ]
  - Checking inetd (telnet)                                  [ NOT FOUND ]

[+] Banners and identification
------------------------------------
  - /etc/motd                                                [ FOUND ]
    - /etc/motd permissions                                  [ OK ]
    - /etc/motd contents                                     [ WEAK ]
  - /etc/issue                                               [ FOUND ]
    - /etc/issue contents                                    [ WEAK ]
  - /etc/issue.net                                           [ FOUND ]
    - /etc/issue.net contents                                [ WEAK ]

[ Press [ENTER] to continue, or [CTRL]+C to stop ]


[+] Scheduled tasks
------------------------------------
  - Checking crontab/cronjob                                 [ DONE ]
  - Checking atd status                                      [ RUNNING ]
    - Checking at users                                      [ DONE ]
    - Checking at jobs                                       [ NONE ]
```

Σχήμα 4.29 Έβδομη οθόνη σάρωσης με το Lynis

Στην επόμενη ενότητα (Insecure services) ο δαίμονας inetd (αναλύεται και ρυθμίζεται σε επόμενο κεφάλαιο) ο οποίος είναι υπεύθυνος για τις υπηρεσίες internet είναι ενεργοποιημένος, αλλά η υπηρεσία telnet είναι ανενεργή.

Στην ενότητα banners and identification το motd το οποίο χρησιμοποιείτε για να αποστέλλει πληροφορίες και προειδοποιήσεις σε όλους τους χρήστες από τον διαχειριστή του συστήματος είναι ενεργό. Μπορούμε να διαμορφώσουμε τις ρυθμίσεις αυτού του αρχείου κάνοντας επεξεργασία το αρχείο etc/motd με τον επεξεργαστεί nano. Για παράδειγμα μπορούμε να τοποθετήσουμε ένα μήνυμα το οποίο θα προειδοποιεί τους χρήστες που εισέρχονται στην υπηρεσία SSH.

Ο crontab (ενότητα scheduled tasks) είναι από τα πιο χρήσιμα εργαλεία που μας δίνει ένα unix σύστημα. Στην ουσία ελέγχει τι θα τρέξει, και ποτέ μειώνοντας έτσι τον φόρτο εργασίας για τους διαχειριστές συστημάτων/εφαρμογών. Ο κάθε χρήστης έχει το δικό του crontab στο όποιο μπορεί να βάλει εντολές η δικά του script να τρέχουν όποτε του πουν η όταν το θέλει αυτός. Φυσικά οι εντολές που μπορεί να τρέξει εξαρτώνται από τα δικαιώματα που έχει στο λειτουργικό σύστημα. Για να μπορέσουμε να επεξεργαστούμε το αρχείο του crontab αρκεί να δώσουμε την παρακάτω εντολή έχοντας δικαιώματα διαχειριστή:

```
crontab -e
```

Μία παρουσίαση του LSM (Linux Security Modules) έγινε στην παράγραφο 3.4. Στο σχήμα 4.30 μπορούμε να δούμε ότι στο σύστημά μας δεν υπάρχει εγκατεστημένο κάποιο εργαλείο LSM. Αυτό δεν είναι κατ' ανάγκη κακό καθώς πολλά από τα εργαλεία αυτά έχουν σχεδιαστεί για σκοπούς που δεν εξυπηρετούν έναν μέσο χρήστη. Το Security Enhanced Linux (SELinux) περιλαμβάνει ένα patch για τον πυρήνα που προσθέτει χαρακτηριστικά ασφαλείας και patches για τις εφαρμογές ώστε να μπορούν να προσδιορίζουν το πεδίο της ασφάλειας στο οποίο τρέχουν οι διεργασίες. Ο κώδικας αναπτύχθηκε αρχικά από την NSA και τελεί υπό την άδεια GPLv2. Η άδεια χρήσης του κώδικα είναι ίδια με αυτή του πυρήνα. Πολλοί ιδιώτες αλλά και εταιρείες έχουν συνεισφέρει στο project (ανάμεσά τους και η NSA). Οι απόψεις για τον συγκεκριμένο εργαλείο ασφαλείας διίστανται καθώς ως βασικός υποστηρικτής του είναι η NSA και πολλοί χρήστες θεωρούν πώς συντρέχουν λόγοι κατασκοπείας που η NSA ασχολείται με την υποστήριξη του συγκεκριμένου εργαλείου. Βέβαια το εργαλείο αυτό είναι ανοικτού κώδικα, το οποίο σημαίνει ότι ο καθένας μπορεί να ελέγξει τον κώδικα του. Ένας από τους προγραμματιστές του Debian GNU / Linux, ο Russell Coker ο οποίος είναι καταχωρημένος στη λίστα ως παράγοντας που συμβάλλει στο SE Linux, ερωτηθείς για το κατά πόσον χρειάζεται ένας έλεγχος κώδικα, είπε στο iTWire:

Ο κώδικας του SE Linux είναι Ελεύθερος για οποιονδήποτε θέλει να τον επανεξετάσει. Ίσως είναι καλύτερα ελεγμένος από τους περισσότερους κώδικες του πυρήνα.

Ένας άλλος προγραμματιστής του Debian ο May, σύμβουλος του open-source, δήλωσε πώς είναι βέβαιος για την ακεραιότητα του κώδικα του SE Linux

Παρόλα αυτά πολλοί χρήστες και hacker ανά τον κόσμο δεν εμπιστεύονται το εν λόγω εργαλείο ασφαλείας. Επίσης μετά από διάφορες ειδήσεις όπως το περίφημο άρθρο: «Η NSA ζήτησε από τον Linux Torvalds να εισάγει κερκόπορτες στα συστήματα GNU/Linux» (http://falkvinge.net) οι χρήστες με το δίκιο του είναι αρκετά καχύποπτοι. Το σίγουρο πάντως είναι πώς όσα εργαλεία ασφαλείας και να χρησιμοποιήσουμε πάντα θα υπάρχει μία δίοδος για τους ενημερωμένους εισβολείς για να εισχωρήσουν στα διάφορα συστήματα. Για το αν το SELinux είναι ασφαλές, μη κατασκοπευτικό και μη κακόβουλο ο έχων γνώση του κώδικα (γραμμένο σε γλώσσα C) μπορεί να τον εξετάσει και να αποφανθεί αν θέλει να το χρησιμοποιήσει η όχι.

```
[+] Security frameworks
------------------------------------
  - Checking presence AppArmor                                [ NOT FOUND ]
  - Checking presence SELinux                                 [ NOT FOUND ]
  - Checking presence grsecurity                              [ NOT FOUND ]
  - Checking for implemented MAC framework                    [ NONE ]

[ Press [ENTER] to continue, or [CTRL]+C to stop ]


[+] Software: file integrity
------------------------------------
  - Checking file integrity tools
  - Checking presence integrity tool                          [ NOT FOUND ]

[ Press [ENTER] to continue, or [CTRL]+C to stop ]


[+] Kernel Hardening
------------------------------------
  - Comparing sysctl key pairs with scan profile
    - kernel.core_uses_pid (exp: 1)                           [ DIFFERENT ]
    - kernel.ctrl-alt-del (exp: 0)                            [ OK ]
    - kernel.kptr_restrict (exp: 1)                           [ DIFFERENT ]
    - kernel.sysrq (exp: 0)                                   [ DIFFERENT ]
    - net.ipv4.conf.all.accept_redirects (exp: 0)             [ DIFFERENT ]
    - net.ipv4.conf.all.accept_source_route (exp: 0)          [ OK ]
    - net.ipv4.conf.all.bootp_relay (exp: 0)                  [ OK ]
    - net.ipv4.conf.all.forwarding (exp: 0)                   [ OK ]
    - net.ipv4.conf.all.log_martians (exp: 1)                 [ DIFFERENT ]
    - net.ipv4.conf.all.mc_forwarding (exp: 0)                [ OK ]
    - net.ipv4.conf.all.proxy_arp (exp: 0)                    [ OK ]
    - net.ipv4.conf.all.rp_filter (exp: 1)                    [ DIFFERENT ]
    - net.ipv4.conf.all.send_redirects (exp: 0)               [ DIFFERENT ]
    - net.ipv4.conf.default.accept_redirects (exp: 0)         [ DIFFERENT ]
    - net.ipv4.conf.default.accept_source_route (exp: 0)      [ DIFFERENT ]
    - net.ipv4.conf.default.log_martians (exp: 1)             [ DIFFERENT ]
    - net.ipv4.icmp_echo_ignore_broadcasts (exp: 1)           [ OK ]
    - net.ipv4.icmp_ignore_bogus_error_responses (exp: 1)     [ OK ]
    - net.ipv4.tcp_syncookies (exp: 1)                        [ OK ]
    - net.ipv4.tcp_timestamps (exp: 0)                        [ DIFFERENT ]
    - net.ipv6.conf.all.accept_redirects (exp: 0)             [ DIFFERENT ]
    - net.ipv6.conf.all.accept_source_route (exp: 0)          [ OK ]
    - net.ipv6.conf.default.accept_redirects (exp: 0)         [ DIFFERENT ]
    - net.ipv6.conf.default.accept_source_route (exp: 0)      [ OK ]
```

Σχήμα 4.30 Όγδοη οθόνη σάρωσης με το Lynis

Στην ενότητα File integrity πληροφορούμαστε για τα πακέτα προγραμμάτων που είναι εγκατεστημένα στο σύστημά μας σχετικά με τον έλεγχο της ακεραιότητας των διαφόρων στοιχείων του συστήματος και σε πραγματικό χρόνο. Τα εργαλεία που μας προτείνει να χρησιμοποιήσουμε το Lynis είναι το AIDE, Samhain και Tripwire.

Ένα εξαιρετικής σημασίας για την ασφάλεια του λειτουργικού μας συστήματος είναι και το αρχείο Sysctl το οποίο περιέχει ρυθμίσεις και παραμέτρους σχετικά με τον πυρήνα του Linux και πώς αυτός να συμπεριφέρεται στις διάφορες καταστάσεις. Το αρχείο αυτό θα πρέπει να τροποποιείτε με ιδιαίτερη προσοχή καθώς

περιέχει ρυθμίσεις οι οποίες μπορούν να επηρεάσουν το σύστημά μας και να το καταστρέψουν κυριολεκτικά.

Για να δούμε τις ρυθμίσεις που υπάρχουν στο αρχείο αυτό αρκεί να δώσουμε την παρακάτω εντολή:

```
sysctl –a
```

Η λίστα αυτή είναι αρκετά εκτενείς και περιέχει πληθώρα ρυθμίσεων που συμβάλουν στην εύρυθμη λειτουργία του πυρήνα και του συστήματος γενικότερα. Για να αλλάξουμε τις τιμές που μας προτείνει το Lynis χρησιμοποιούμε την παρακάτω εντολή ακολουθούμενη από την λειτουργία που θέλουμε να αλλάξουμε:

```
sysctl –w kernel.core_uses_pid = 1
```

Η παραπάνω εντολή αλλάζει την τιμή τις ρύθμισης `kernel.core_uses_pid` από 0 σε 1. Η εντολή αυτή είναι υπεύθυνη για τις απορρίψεις διεργασιών του πυρήνα. Οι απορρίψεις του πυρήνα χρησιμοποιούνται συχνά για τη διάγνωση ή τον εντοπισμό σφαλμάτων στα προγράμματα του Linux ή UNIX. Οι απορρίψεις πυρήνα μπορούν να χρησιμεύσουν ως χρήσιμα βοηθήματα εργαλεία εντοπισμού σφαλμάτων για admins έτσι ώστε να μάθουν γιατί μία εφαρμογή όπως τα Lighttpd, Apache, PHP-CGI ή οποιοδήποτε άλλο πρόγραμμα συνετρίβη.

Οι ρυθμίσεις των παραμέτρων του πυρήνα διαφέρουν ανάλογα με τον χρήστη. Για παράδειγμα η παραπάνω εντολή για έναν απλό χρήστη δεν έχει καμία σημασία καθώς δεν πρόκειται ποτέ να ασχοληθεί με το ζήτημα του εντοπισμού και τις λύσεις των διαφόρων σφαλμάτων που μπορεί να υπάρχουν στο σύστημά του. Αντίθετα για τους διαχειριστές συστημάτων παρόμοιες εντολές όπως η παραπάνω είναι εξαιρετικής σημασίας. Κάποιες σημαντικές ρυθμίσεις που πρέπει να γίνουν με τον τρόπο που υποδείχθηκε παραπάνω είναι οι εξής:

`net.ipv4.tcp_syncookies = 1` (αποτροπή επιθέσεων DDoS)

`net.ipv4.icmp_echo_ignore_broadcasts = 1` (αποτροπή αιτημάτων ping)

`net.ipv4.conf.all.rp_filter = 1` (προστασία ενάντια πλαστογραφημένων διευθύνσεων ip)

4.5.2 Τα εργαλεία chkrootkit και Rkhunter

Δύο πολύ χρήσιμα εργαλεία για τα συστήματα του Unix και του Linux είναι τα chkrootkit και Rkhunter τα οποία κάνουν ουσιαστικά το ίδιο πράγμα, ελέγχουν το σύστημά μας για rootkits. Πολλοί χρήστες υποστηρίζουν πώς η χρησιμοποίηση και των δύο εργαλείων είναι απαραίτητη διότι μπορεί το πρώτο εργαλείο (chkrootkit) να μην εντοπίσει κάποιο rootkit ή να εντοπίσει κάποιο εσφαλμένα. Αρκετά bugs έχουν αναφερθεί για το πρώτο εργαλείο με αποτέλεσμα πολλές φορές να αναφέρει

λανθασμένα αποτελέσματα. Οπότε η χρήση και των δύο εργαλείων είναι απαραίτητη. Προφανώς, ένας εισβολέας μπορεί εύκολα να τροποποιήσει τις πηγές rootkit και να αλλάξει την υπογραφή του, ώστε να αποφύγει τον εντοπισμό. Αν το chkrootkit δε μπορεί να βρεί γνωστή υπογραφή μέσα σε ένα αρχείο, δεν έχει τη δυνατότητα να προσδιορίσει αυτόματα αν έχει μολυνθεί από trojan. Σ' αυτή την περίπτωση μπορούμε να δοκιμάσουμε να εκτελέσουμε την εφαρμογή σε κατάσταση έμπειρου (προχωρημένου) χρήστη (επιλογή-x). Σε αυτή τη λειτουργία ο χρήστης μπορεί να εξετάσει ύποπτα κλειδιά που μπορούν να “ξεσκεπάσουν” ένα trojan.

Η εγκατάσταση των δύο αυτών εργαλείων γίνεται ακριβώς με τον ίδιο τρόπο όπως και με το εργαλείο Lynis (εγκατάσταση του Lynis). Μπορούμε να κατεβάσουμε τα δύο αυτά εργαλεία από τους παρακάτω συνδέσμους:

http://packages.slackonly.com/pub/packages/14.1-x86_64/system/chkrootkit/
http://slackware.uk/slacky/slackware-13.37/security/rkhunter/1.4.0/

Για να ελέγξουμε το σύστημά μας για παρουσία rootkit με το εργαλείο chkrootkit εκτελούμε την παρακάτω εντολή:

```
chkrootkit
```

Για να ελέγξουμε το σύστημά μας για παρουσία rootkit με το εργαλείο Rkhunter εκτελούμε την παρακάτω εντολή:

```
rkhunter --check
```

4.6 Απενεργοποίηση και αλλαγή ρυθμίσεων στις υπηρεσίες που εκτελούνται

Σημαντικό κομμάτι στην ασφάλεια του συστήματός μας αποτελεί και η απενεργοποίηση των διαφόρων υπηρεσιών που δεν χρησιμοποιούμαι και δεν χρειαζόμαστε. Όσο λιγότερες υπηρεσίες εκκινούνται και εκτελούνται τόσο λιγότερο ευάλωτοι είμαστε σε επιθέσεις. Οι υπηρεσίες εκκινούνται κυρίως από δύο σημεία:

1. το inetd
2. και διάφορα script εκκίνησης

Αρκετές υπηρεσίες εκκινούνται από το inetd στο slackware. Το inetd είναι ένας δαίμονας του συστήματος ο οποίος όπως είχαμε εξηγήσει και στο κεφάλαιο 2 εκκινεί και παρέχει υπηρεσίες internet (ορισμένες εκδόσεις Linux χρησιμοποιούν και τον xinetd δαίμονα). Κάποιες υπηρεσίες που παρέχονται από τον δαίμονα αυτό θα πρέπει να απενεργοποιούνται για λόγους ασφαλείας. Μία υπηρεσία που θα πρέπει να απενεργοποιήσουμε είναι η telnet η οποία δεν χρησιμοποιείται πλέον εκτός από επιχειρήσεις οι οποίες την χρησιμοποιούν για τον έλεγχο άλλων πρωτοκόλλων, όπως το SMTP, το POP3 κτλ. Χρησιμοποιείται επίσης πολύ από μεγάλους υπολογιστές (mainframes) εκπαιδευτικών ιδρυμάτων, μεγάλων εταιρειών και παρόμοιων φορέων, επειδή, παρέχει εξαιρετικές δυνατότητες ελέγχου στα επιμέρους στοιχεία ενός δικτύου αν και πλέον έχει αντικατασταθεί από το ssh, scp κλπ. Αρκετά συστήματα Linux έχουν πια απενεργοποιημένη εκ των προτέρων την συγκεκριμένη υπηρεσία, όμως καλό είναι να γίνεται ένας έλεγχος για τις υπηρεσίες που είναι ενεργοποιημένες στο αρχείο inetd. Η απενεργοποίηση μιας υπηρεσίας μπορεί να γίνει με την προσθήκη της # στην αρχή της εντολής που επιθυμούμαι να απενεργοποιήσουμε. Έτσι για παράδειγμα για να ανοίξουμε το αρχείο inetd.conf εκτελούμε την παρακάτω εντολή στο τερματικό μας έχοντας δικαιώματα διαχειριστή:

```
nano /etc/inetd.conf
```

Το nano είναι ένας editor ο οποίος μπορεί να μας βοηθήσει στην επεξεργασία του συγκεκριμένου αρχείου ρυθμίσεων. Εντοπίζουμε στην συνέχεια την εντολή που θέλουμε να απενεργοποιήσουμε και προσθέτουμε τη δίεση στην αρχή της εντολής. Στο δικό μας παράδειγμα απενεργοποιούμαι την υπηρεσία telnet που την βρίσκουμε με την παρακάτω μορφή:

```
#telnet stream tcp nowait root /usr/sbin/tcpd in.telnetd
```

Για να αποθηκεύσουμε το αρχείο πρέπει να πατήσουμε το πλήκτρο F3 και στην συνέχεια να αποδεχτούμε τις αλλαγές. Με το πλήκτρο F2 μπορούμε να εξέλθουμε. Η υπηρεσία θα απενεργοποιηθεί μετά την επανεκκίνηση του inetd.

Οι υπηρεσίες που εκκινούνται από τα διάφορα scripts βρίσκονται στο `/etc/rc.d/`. Σε αυτή τη διαδρομή βρίσκεται και το SSH (Secure Shell) το οποίο είναι ένα (ασφαλές) δικτυακό πρωτόκολλο το οποίο επιτρέπει τη μεταφορά δεδομένων μεταξύ δύο υπολογιστών. Το συγκεκριμένο πρωτόκολλο έχει δεχτεί

αρκετές επιθέσεις και εξακολουθεί να δέχεται αρκετές επιθέσεις ωμής βίας με απώτερο στόχο την συλλογή των ευαίσθητων πληροφοριών και απώτερο στόχο την είσοδο των εισβολέων στο σύστημά μας. Σε αντίθεση βέβαια με το telnet το οποίο δεν κρυπτογραφεί τίποτα από ότι στέλνουμε το SSH παρέχει περισσότερη ασφάλεια και προστασία. Το να απενεργοποιήσουμε εντελώς το SSH το οποίο πολλοί υποστηρίζουν ότι είναι ένας καλός τρόπος προφύλαξης από τις διάφορες επιθέσεις brute force (ωμής βίας) δεν μπορεί να θεωρηθεί ως μέτρο προφύλαξης. Αντίθετα αυτό που μπορούμε να κάνουμε είναι κάποιες παραμετροποιήσεις στην υπηρεσία και στον δαίμονα αυτής έτσι ώστε να είναι δυσκολότερο σε κάποιον εισβολέα να εισέλθει στο σύστημά μας.

Το αρχείο ρυθμίσεων του SSH βρίσκεται στην διαδρομή `/etc/ssh/sshd_conf.` Από προεπιλογή το SSH ακούει στην πόρτα 22. Οι εισβολείς χρησιμοποιούν λογισμικό το οποίο κατασκοπεύει τις πόρτες και δίνει πληροφορίες για το αν ο συγκεκριμένος χρήστης χρησιμοποιεί το SSH. Είναι λοιπόν επιβεβλημένο να αλλάξουμε την πόρτα αυτή με μία η οποία να είναι πάνω από το 1024 και αυτό διότι πολλά προγράμματα κατασκοπείας πορτών (συμπεριλαμβανομένου και του nmap) δεν σαρώνουν πόρτες μεγαλύτερες από αυτόν τον αριθμό. Για να το κάνουμε λοιπόν αυτό εκτελούμε την παρακάτω εντολή στο τερματικό μας με δικαιώματα διαχειριστή:

```
nano /etc/ssh/sshd_config
```

Εντοπίζουμε την γραμμή που αναφέρεται στην πόρτα 22 (port 22) και αντικαθιστούμε τον αριθμό αυτό με κάποια πόρτα μεγαλύτερη του 1024. Προσέχουμε επίσης να μην βάλουμε κάποια η οποία χρησιμοποιείτε από άλλη υπηρεσία. Μία καλή επιλογή είναι η πόρτα 1196. Για να δούμε επίσης τις ανοιχτές πόρτες στο σύστημά μας μπορούμε να εκτελέσουμε την παρακάτω εντολή:

```
Netstat -tulpn
```

Το SSH έχει επίσης δύο εκδόσεις που μπορούμε να χρησιμοποιήσουμε η πρώτη είναι η προεπιλεγμένη και ονομάζεται SSH protocol 1 και η δεύτερη SSH protocol 2. Το πρώτο πρωτόκολλο SSH υπόκειται σε κάποια θέματα ασφάλειας, συμπεριλαμβανομένων των man-in-the-middle επιθέσεων και τις απόκτησης δικαιωμάτων διαχειριστή εξ αποστάσεως. Συνίσταστε να χρησιμοποιήσουμε το πρωτόκολλο 2 το οποίο είναι λιγότερο επιρρεπές σε τέτοιου είδους επιθέσεις. Για να το κάνουμε αυτό εντοπίζουμε τη γραμμή `Protocol 2,1` και διαγράφουμε τον αριθμό 1 έτσι ώστε να υπάρχει μόνο ο αριθμός 2 χωρίς το κόμμα.

Δεν θα πρέπει επίσης να επιτρέπονται συνδέσεις του διαχειριστή μέσω SSH, γιατί αυτό αποτελεί έναν μεγάλο κίνδυνο για την ασφάλεια του συστήματος. Εάν ένας εισβολέας αποκτήσει δικαιώματα διαχειριστή στο σύστημά μας, μπορεί να κάνει μεγαλύτερη ζημιά από ό, τι αν αποκτούσε δικαιώματα απλού χρήστη. Γι αυτό θα πρέπει να διαμορφώσουμε το SSH έτσι ώστε στον χρήστη root να μην επιτρέπεται να συνδέεται. Βρίσκουμε λοιπόν τη γραμμή που λέει.:

```
PermitRootLogin yes
```

και έπειτα αλλάζουμε το yes σε no.

4.7 Τακτικός έλεγχος του αρχείου passwd

Για να είμαστε περισσότερο ασφαλείς καλό είναι να ελέγχουμε το αρχείο που βρίσκεται στην διαδρομή `/etc/passwd`. Αυτό το αρχείο έχει όλα τα στοιχεία των λογαριασμών, σε πεδία που διαχωρίζονται με άνω και κάτω τελεία: Το πρώτο πεδίο είναι το όνομα της σύνδεσης, το δεύτερο πεδίο είναι ο κωδικός πρόσβασης, και το τρίτο πεδίο είναι ο αριθμός id χρήστη. Εάν το δεύτερο πεδίο είναι κενό, ::, αυτό σημαίνει ότι δεν υπάρχει κωδικός πρόσβασης για το λογαριασμό αυτό και ο καθένας μπορεί να συνδεθεί. Δεν πρέπει να υπάρχουν λογαριασμοί χωρίς κωδικό πρόσβασης για προφανείς λόγους. Ένας αστερίσκος * στο πεδίο του κωδικού σημαίνει ότι κανείς δεν μπορεί να συνδεθεί. Αν στο τρίτο πεδίο, το αναγνωριστικό χρήστη (UID) είναι 0 τότε αυτός ο λογαριασμός έχει πλήρεις εξουσίες διαχειριστή. Οι εισβολείς που θέλουν να έχουν πρόσβαση στον υπολογιστή μας μπορεί να προσθέσουν μια καταχώρηση διαχειριστικών δικαιωμάτων (`root`) στο τέλος του αρχείου `/etc/passwd`.

4.8 Παραμετροποίηση του αρχείου /boot έτσι ώστε να είναι μόνο για ανάγνωση

Ο kernel του Linux και τα σχετικά αρχεία του βρίσκονται στον κατάλογο /boot ώς εξ ορισμού είναι και για ανάγνωση αλλά και για εγγραφή. Η αλλαγή της πολιτικής σε μόνο για ανάγνωση, μειώνει τον κίνδυνο από μη εξουσιοδοτημένες τροποποιήσεις των σημαντικών αρχείων εκκίνησης. Για να το κάνουμε αυτό, ανοίγουμε το /etc/fstab αρχείο με την παρακάτω εντολή έχοντας δικαιώματα διαχειριστή

```
nano /etc/fstab
```

αφού το ανοίξουμε στο τέλος του αρχείου προσθέτουμε την παρακάτω εντολή:

```
LABEL=/boot      /boot      ext4      defaults,ro      1 2
```

Εάν στο μέλλον επιθυμούμαι να ανανεώσουμε τον πυρήνα του Linux θα πρέπει να επεξεργαστούμε ξανά το αρχείο αυτό και να διαγράψουμε την εντολή που δηλώσαμε στο τέλος του αρχείου.

4.9 Στατιστικά στοιχεία και παρακολούθηση του δικτύου με το εργαλείο iptraf

Ένα πολύ χρήσιμο εργαλείο που έρχεται προεγκατεστημένο με το slackware είναι το iptraf το οποίο μας παρέχει αρκετές δυνατότητες σχετικά με τα διάφορα στατιστικά στοιχεία του δικτύου, συμπεριλαμβανομένων των πληροφοριών για τα πρωτόκολλα TCP, UDP, ICMP και OSPF, πληροφορίες φορτίου Ethernet, στατιστικά

κόμβων, λάθη checksum της IP, και άλλα. Μπορεί να παρέχει τις ακόλουθες πληροφορίες:

Στατιστικά στοιχεία κίνησης του δικτύου για το TCP
Στατιστικά στοιχεία κίνησης των IP
Στατιστικά στοιχεία κίνησης δικτύου με βάση τα πρωτόκολα
Στατιστικά στοιχεία κίνησης με βάση τις θύρες των UDP/TCP και με βάση το μέγεθος των πακέτων κ.α.

4.10 Ρύθμιση του iptables

Ο πυρήνας του Linux προσφέρει ένα πολύ ευέλικτο και δυνατό σύστημα (framework) φιλτραρίσματος πακέτων με όνομα netfilter, το σύστημα αυτό ελέγχεται από το πρόγραμμα iptables.

Ο iptables είναι μία εφαρμογή χρήστη που επιτρέπει σε ένα διαχειριστή να ρυθμίζει τους πίνακες που παρέχονται από το τείχος προστασίας του πυρήνα του Linux και τις δεσμεύσεις και κανόνες που περιλαμβάνονται σε αυτόν. Διάφορα modules και προγράμματα του πυρήνα χρησιμοποιούνται σήμερα για διαφορετικά πρωτόκολλα. Ο iptables ισχύει για IPv4, ip6tables σε IPv6, arptables για ARP, και ebtables για το Ethernet.

Ο iptables απαιτεί αυξημένα δικαιώματα για να λειτουργήσει και πρέπει να εκτελείται από τον root χρήστη. Στα περισσότερα συστήματα Linux καθώς και στο slackware, ο iptables εγκαθίσταται στην διαδρομή /usr/sbin/iptables.

Ο όρος iptables συνήθως χρησιμοποιείται για να αναφερθεί εξ ολοκλήρου στα συστατικά στοιχεία του πυρήνα. Ο διάδοχος του iptables είναι ο nftables, ο οποίος συγχωνεύθηκε με τον πυρήνα του Linux στην έκδοση 3.13, η οποία κυκλοφόρησε στις 19 Ιανουαρίου του 2014

Στην έκδοση 14.1 (που είναι και η τελευταία σταθερή έκδοση) του slackware που κυκλοφόρησε το 2013 ο πυρήνας του Linux είναι ο 3.10.17 και ο nftables δεν υπάρχει προεγκατεστημένος. Θα πρέπει να τον εγκαταστήσουμε προκειμένου να τον χρησιμοποιήσουμε. Στα πλαίσια της παρούσας διπλωματικής όμως θα κάνουμε χρήση του ήδη προεγκατεστημένου iptables.

Καθώς απαιτείτε να γνωρίζουμε σε βάθος την σύνταξη των εντολών για να χρησιμοποιήσουμε τον iptables τα παρακάτω σχόλια και διευκρινήσεις είναι προς βοήθεια για το τι κάνει η κάθε εντολή.

Η βασική σύνταξη της εντολής iptables είναι η ακόλουθη:

```
iptables [-t table-name] command chain-name rule-description
```

- ✓ Η παράμετρος table-name ορίζει τον πίνακα με τον οποίο θα ασχοληθούμε και παίρνει τις τιμές filter (είναι ο πίνακας στον οποίο γίνεται το firewalling), mangle (ο πίνακας στον οποίο πειράζονται τα διάφορα πακέτα). Αν παραληφθεί, τότε ο iptables υποθέτει ότι μιλάμε για το filter table.
- ✓ Η command ορίζει τί θέλουμε να κάνουμε με μια αλυσίδα (πρόκειται για τις εντολές INPUT, OUTPUT, FORWARD κλπ).

- ✓ chain-name είναι το όνομα της αλυσίδας
- ✓ και ακολουθεί η περιγραφή του κανόνα, με τα κριτήρια επιλογής των πακέτων και την ενέργεια που θα ληφθεί για τα πακέτα που ταιριάζουν.

Οι κυριότερες διαθέσιμες εντολές είναι:

- ✓ Προσθήκη κανόνα στο τέλος της αλυσίδας: iptables *-A INPUT -j LOG.*
- ✓ Προσθήκη κανόνα σε προκαθορισμένη θέση της αλυσίδας, πριν από τον κανόνα που υπήρχε εκεί: ipta **3** -p tcp --dport ssh -j ACCEPT
- ✓ Αλλαγή κανόνα σε συγκεκριμένη θέση της αλυσίδας: iptables *-R INPUT 3 -p* tcp --dport http -j ACCE .
- ✓ Διαγραφή κανόνα από συγκεκριμένη θέση της αλυσίδας: iptables *-D INPUT 3*
- ✓ Διαγραφή όλων των κανόνων της αλυσίδας: iptables *-F INPUT.*
- ✓ Διαγραφή όλων των κανόνων όλων των αλυσίδων όλου του πίνακα: iptables -t nat –F.
- ✓ Καθορισμός default policy κάποιας αλυσίδας: iptables *-P INPUT DROP* .
- ✓ Δημιουργία user-defined αλυσίδας: iptables *-N a_new_chain.*
- ✓ Εμφάνιση των κανόνων που περιέχει μια αλυσίδα: iptables *-L chain_name (-v)*
- ✓ Εμφάνιση όλων των κανόνων όλων των αλυσίδων ενός πίνακα: iptables -t nat-L

Η περιγραφή του κανόνα ξεκινά με τα κριτήρια που πρέπει να πληρεί το πακέτο:

- ✓ Πρωτόκολλο: -p [tcp,icmp,udp,...]
- ✓ Διεύθυνση αποστολέα: -s 1.2.3.4
- ✓ Διεύθυνση παραλήπτη: -d 2.3.4.5
- ✓ Θύρα αποστολής: --sport 123 ή --source-port 1234
- ✓ Θύρα προορισμού: --dport 2345 ή --destination-port 2345
- ✓ Interface εισόδου (μόνο στις *INPUT, PREROUTING, FORWARD): -i eth0*
- ✓ Interface εξόδου (μόνο στις *OUTPUT, POSTROUTING, FORWARD*): -o eth1

και ολοκληρώνεται με την ενέργεια-προορισμό: -j [*ACCEPT, DROP, REJECT, LOG, ...].*

Για να μπορέσουμε να χρησιμοποιήσουμε τον iptables αρκεί να δημιουργήσουμε ένα εκτελέσιμο αρχείο με τους κανόνες που θέλουμε. Η διαδικασία έχει ως εξής:

Δημιουργούμε το αρχείο `rc.firewall` στην διαδρομή *`/etc/rc.d/rc.firewall`* με το nano. Για να το κάνουμε αυτό αρκεί αν πληκτρολογήσουμε την παρακάτω εντολή:

```
nano /etc/rc.d/rc.firewall
```

Στο τερματικό μας θα ανοίξει ένα κενό αρχείο με όνομα `rc.firewall`. Στο αρχείο αυτό μπορούμε να προσθέσουμε τους κανόνες που χρειαζόμαστε.

Οι παρακάτω εντολές αποτελούν μερικά παραδείγματα τα οποία μπορούν να χρησιμοποιηθούν για συγκεκριμένους σκοπούς, ανάλογα με τις ανάγκες που τα

χρειάζεται κάποιος. Για παράδειγμα διαφορετικές ρυθμίσεις θα πρέπει να κάνει ένας χρήστης στο σύστημα που χρησιμοποιεί στο σπίτι του και άλλες σε έναν Server.

Η εντολή *#!/bin/sh* μπαίνει στην αρχή διότι ορίζει πώς καταρχήν πρόκειται για ένα script, κατά δεύτερον πώς όλες οι εντολές είναι ένα ενιαίο σύνολο εντολών και όχι ξεχωριστές μεταξύ τους και τρίτον πώς να εκτελεστούν οι παρακάτω εντολές (ο διερμηνέας θα είναι το sh). Με την δίεση υποδηλώνονται τα σχόλια. Στο Unix η δίεση που ακολουθείτε από θαυμαστικό ονομάζεται shebang (#!).

```
#!/bin/sh

# Αν θέλουμε να παρακάμψουμε τις τροποποιήσεις που κάναμε στον πυρήνα με την
# βοήθεια του Lynis, μπορούμε να το κάνουμε με την παρακάτω εντολή
SYSCTL="/sbin/sysctl -w"

# Καθορισμός της διαδρομής για τον iptables και ονομασία για περισσότερη ευκολία
# στη χρησιμοποίησή της μέσα στους κανόνες.
iptables="/usr/sbin/iptables"
IPTS="/usr/sbin/iptables-save"
IPTR="/usr/sbin/iptables-restore"

# Internet Interface. Ανάλογα με το ποια διεπαφή χρησιμοποιούμε. Στην
# συγκεκριμένη περίπτωση κάνουμε χρήση μέσω ethernet
INET_IFACE="eth0"

# Η παρακάτω εντολή μας προστατεύει από ενδεχόμενες DoS επιθέσεις. Στην
# παρακάτω εντολή ορίζουμε το σύστημά μας να δέχεται 25 αιτήσεις το λεπτό.
# Ενδείκνυται για Severs
iptables -A INPUT -p tcp --dport 80 -m limit --limit
25/minute --limit-burst 100 -j ACCEPT

# Η παρακάτω εντολή ενεργοποιεί την προστασία από επιθέσεις SYN flood.
# Η επίθεση SYN flood είναι ένα είδος επίθεσης άρνησης πρόσβασης (DOS – Denial
# of Service) κατά την οποία ο επιτιθέμενος αποστέλλει πολλαπλές αιτήσεις SYN
# προς το θύμα. Ενδείκνυται για Severs.
if [ "$SYSCTL" = "" ]
then
    echo "1" > /proc/sys/net/ipv4/tcp_syncookies
else
    $SYSCTL net.ipv4.tcp_syncookies="1"
fi

# Επιτρέπουμε 5 ping/s, με ανοχή για 10 μαζεμένα. Αυτό μας προστατεύει από ένα
# ενδεχόμενο pingflow.
iptables -p icmp --icmp-type ! echo-request -j ACCEPT
```

Μεταμφίεση των εσωτερικών ip έτσι ώστε να μην είναι ορατοί στο εξωτερικό
δίκτυο μέσω του NAT
```
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

Αν στο δίκτυο μας χρησιμοποιούμε διασυνδέσεις μέσω τον eth1,2,3 (NICS) τότε
μπορούμε να τις αποδεχόμαστε ή να τις απορρίπτουμε μέσω τον παρακάτω
εντολών. Η απόρριψη μπορεί να γίνει με την εντολή *DROP*
```
iptables -A FORWARD -i eth1 -j ACCEPT
iptables -A FORWARD -i eth2 -j ACCEPT
iptables -A FORWARD -i eth3 -j ACCEPT
```

Αποδοχή των πολιτικών εξόδου. Η απόρριψη μπορεί να γίνει με την εντολή *DROP*
```
iptables -P OUTPUT ACCEPT
```

Αποδοχή των πολιτικών εισόδου. Η απόρριψη μπορεί να γίνει με την εντολή *DROP*
```
iptables -P INPUT ACCEPT
```

Άνοιγμα των πορτών/θυρών 21 και 22 για τα FTP και SSH
Μπορούμε επίσης να αντικαταστήσουμε με την πόρτα 1196 που είχαμε αλλάξει
```
iptables -A INPUT -p tcp --destination-port 1:20 -i eth0
-j DROP
```

Άνοιγμα τις θύρας 25 για των SMTP server
```
iptables -A INPUT -p tcp --destination-port 23:24 -i eth0
-j DROP
```

Άνοιγμα τις θύρας 80 για HTTP server
```
iptables -A INPUT -p tcp --destination-port 26:79 -i eth0
-j DROP
```

Άνοιγμα τις θύρας 110 για POP3 server
```
iptables -A INPUT -p tcp --destination-port 81:109 -i
eth0 -j DROP
```

Άνοιγμα τις θύρας 3306 για MySQL server
```
iptables -A INPUT -p tcp --destination-port 111:3305 -i
eth0 -j DROP
```

Μία καλή πρακτική αφού ανοίξουμε τις θύρες που χρειαζόμαστε είναι να
κλείσουμε όλες τις υπόλοιπες
```
iptables -A INPUT -p tcp --destination-port 3307:65535 -i
eth0 -j DROP
```

Κλείσιμο των πορτών Netbios/SMB UDP
```
iptables -A INPUT -p udp --destination-port 130:145 -i
eth0 -j DROP
```

Η παραπάνω λίστα δεν εξαλείφει όλες τις πιθανές επιθέσεις που μπορεί να μας συμβούν. Είναι παραδείγματα τα οποία είναι χρήσιμα να γνωρίζουμε για το πώς μπορούμε να χειριστούμε τους κανόνες του iptables και μπορεί να μας προστατέψουν σε περιπτώσεις γνωστών επιθέσεων.

Στη συνέχεια για να μπορέσουμε να εκτελέσουμε τους κανόνες που εισάγαμε, θα πρέπει να δώσουμε την παρακάτω εντολή:

```
Chmod +x /etc/rc.d/rc.firewall
```

Για να είναι δυνατή επίσης η λειτουργία του συγκεκριμένου script κατά την εκκίνηση θα πρέπει να καταχωρίσουμε τις παρακάτω εντολές στο αρχείο που βρίσκετε στην διαδρομή /etc/rc.d/rc.M

```
if [ -x /etc/rc.d/rc.firewall ]; then
   /etc/rc.d/rc.firewall
fi
```

4.11 Σύντομη αναφορά στην διανομή tales

Μία νέα διανομή που χρήζει προσοχής είναι η tales η οποία βασίζεται στο debian. Η tales είναι μία live διανομή και δεν απαιτεί εγκατάσταση. Αυτό γίνεται κυρίως έτσι ώστε όποιες εργασίες και αν γίνουν κατά την διάρκεια τις εκτέλεσής της να μην αποθηκεύονται. Ουσιαστικά πρόκειται για μία διανομή που ''ξεχνάει'' ότι κάνει ο χρήστης της επιστρέφοντας στο αρχικό στάδιο έτσι όπως είχε δημιουργηθεί.

Σκοπός της είναι η προστασία της ιδιωτικής ζωής και η ανωνυμία.

Βοηθά έτσι ώστε:

- ✓ Να χρησιμοποιούμε το διαδίκτυο ανώνυμα
- ✓ Όλες οι συνδέσεις με το διαδίκτυο αναγκαστικά να περνούν μέσα από το δίκτυο Tor
- ✓ Να μην αφήνει ίχνη στον υπολογιστή που χρησιμοποιούμε

Χρησιμοποιεί εργαλεία κρυπτογράφησης για την κρυπτογράφηση των αρχείων, e-mail και των άμεσων μηνυμάτων.

Το **Tor** (συντομογραφία του **The onion router**) είναι ένα σύστημα που δίνει στους χρήστες του τη δυνατότητα ανωνυμίας στο Διαδίκτυο. Το λογισμικό πελάτη Tor δρομολογεί τη διαδικτυακή κίνηση μέσω ενός παγκόσμιου εθελοντικού δικτύου διακομιστών με σκοπό να αποκρύψει την τοποθεσία ενός χρήστη ή τη χρήση της κίνησης από οποιονδήποτε διεξάγει διαδικτυακή παρακολούθηση ή ανάλυση της διαδικτυακής κίνησης. Η χρήση Tor κάνει δύσκολη την ανίχνευση διαδικτυακής δραστηριότητας του χρήστη, συμπεριλαμβανομένου επισκέψεων σε κάποια ιστοσελίδα, διαδικτυακές αναρτήσεις, προγράμματα άμεσων μηνυμάτων και άλλων μέσων διαδικτυακής επικοινωνίας, κι έχει σκοπό να προστατεύσει την ατομική

ελευθερία, την ιδιωτικότητα και τη δυνατότητα του χρήστη να διεξάγει εμπιστευτικές εργασίες χωρίς να καταγράφονται οι διαδικτυακές δραστηριότητές του. Το “Onion routing” αναφέρεται στη στρωματοποιημένη φύση της υπηρεσίας κρυπτογράφησης: τα αρχικά δεδομένα κρυπτογραφούνται και επανακρυπτογραφούνται πολλές φορές, έπειτα στέλνονται μέσω διαδοχικών κόμβων του Tor, ο καθένας από τους οποίους αποκρυπτογραφεί ένα «στρώμα» κρυπτογράφησης προτού μεταφέρει τα δεδομένα στον επόμενο κόμβο και τελικά στον προορισμό τους. Αυτό μειώνει την πιθανότητα να αποκρυπτογραφηθούν ή να γίνουν κατανοητά κατά τη μεταφορά τους τα αρχικά δεδομένα. Το Tor είναι ελεύθερο λογισμικό πελάτη και η χρήση του είναι δωρεάν.

Σύνοψη, συμπεράσματα και μελλοντικές επεκτάσεις

Είδαμε λοιπόν κατά την διάρκεια της εκπόνησης, όλα εκείνα τα στοιχεία που απαρτίζουν ένα λειτουργικό σύστημα και βάλαμε τις βάσεις για την περαιτέρω κατανόησή τους. Αναλύθηκαν διεξοδικά τα δύο κύρια λειτουργικά συστήματα τα οποία αποτέλεσαν και αποτελούν βάση για την δημιουργία όλων των υπολοίπων. Αναφερθήκαμε στους πυρήνες, στα συστήματα αρχείων, στα διάφορα γραφικά περιβάλλοντα και τις διανομές. Δόθηκαν απαντήσεις σε ερωτήματα τα οποία είχαμε θέσει εξαρχής όπως το τι είναι ένα λειτουργικό σύστημα, από τι αποτελείται, αν μπορούμε να είμαστε ασφαλείς και ποιες ενέργειες πρέπει να κάνουμε ούτως ώστε να μας παρέχει την απαιτούμενη ασφάλεια που θέλουμε. Πραγματοποιήσαμε μία πλήρη εγκατάσταση της διανομής Slackware Linux που αποτελεί την παλαιότερη εν ενεργεία από θέματα συντήρησης διανομή και δώσαμε τις απαραίτητες κατευθύνσεις για την ρύθμιση και αποτροπή όλων εκείνων των στοιχείων που μπορούν να βλάψουν το σύστημά μας. Έχοντας βέβαια υπόψη το μεγάλο εύρος θεμάτων που διερεύνησε η παρούσα διπλωματική και το γεγονός ότι οι μεγαλύτερες εταιρείες και κυβερνήσεις στον κόσμο ασχολούνται με την ασφάλεια των λειτουργικών συστημάτων και δαπανούν χιλιάδες ανθρωποώρες με σκοπό την παροχή αξιόπιστων λύσεων ασφαλείας, η εκτενέστερη έρευνα κρίνεται επιτακτική. Οι βελτιώσεις που μπορούν να γίνουν στο σύστημά μας ως προς το θέμα της ασφάλειας δεν μπορούν να περιοριστούν σε στενά πλαίσια και θα λέγαμε ότι μπορεί να είναι τόσα όσα και τα στοιχεία που απαρτίζουν το λειτουργικό μας σύστημα. Ωστόσο πρέπει να ληφθεί υπόψη ότι όσο η εξάρτηση των επιχειρήσεων και χρηστών από το διαδίκτυο αυξάνεται και τα προγράμματα προστασίας βελτιώνονται, τόσο οι κυβερνο-εγκληματίες γίνονται καλύτεροι. Εκμεταλλεύονται τα τρωτά σημεία των εφαρμογών και συνιστωσών του λειτουργικού συστήματος, αναζητούν καινούριους δρόμους στη διακίνηση παράνομου λογισμικού με απώτερο στόχο το οικονομικό όφελος. Συνεχώς εξελίσσονται οι επιθέσεις και γίνονται παραλλαγές τους με σκοπό τον μη εντοπισμό τους. Για την προστασία από όλους τους κινδύνους, απαιτείται πολυεπίπεδη προσέγγιση των θεμάτων ασφαλείας, κάτι το οποίο απαιτεί εξειδικευμένη έρευνα και γνώση σε βάθος όλων των θεμάτων της επιστήμης της πληροφορικής. Πάντοτε πρέπει επίσης να λαμβάνουμε υπόψη μας ότι όσα μέτρα ασφαλείας και αν πάρουμε, όσο καλά και αν θωρακίσουμε το σύστημά μας πάντοτε θα υπάρχει κάπου κάποιο μικρό παραθυράκι μέσω του οποίου ο εισβολέας θα μπορέσει να εισχωρήσει στο σύστημά μας.

Αναφορές

Ελληνική Βιβλιογραφία

Βασιλάκης Κ., Προστασία και ασφάλεια υπολογιστικών συστημάτων, Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών, τμήμα πληροφορικής και τηλεπικοινωνιών, 2005

Ευσταθόπουλος Πέτρος, Σχεδιασμός κατανεμημένου μοιραζόμενου συστήματος αρχείων στη μνήμη συστοιχίας υπολογιστών, Εθνικό Μετσόβιο Πολυτεχνείο, 2001

Κατσούλας Ν., Όροβας Χ., Παναγιωτίδης Σ., Λειτουργικά συστήματα και ασφάλεια πληροφοριακών συστημάτων, Ινστιτούτο Τεχνολογίας Υπολογιστών & Εκδόσεων «Διόφαντος»

Κοκκινόπουλος Χρήστος, Ηλίας Κ. Σάββας, Τεχνολογικό Εκπαιδευτικό Ίδρυμα Λάρισας, Προγραμματισμός Ι (γλώσσα Προγραμματισμού C), 2005

Κούκης Ευάγγελος, Υλοποίηση ενός Γενικού Στρώματος Επικοινωνίας Πυρήνα προς Πυρήνα Linux μέσω Χώρου Χρήστη, Χρησιμοποιώντας Δίκτυα SCI και FastEthernet, Εθνικό Μετσόβιο Πολυτεχνείο, 2002

Σπυράκης Παύλος, Ελληνικό Ανοικτό Πανεπιστήμιο, Λειτουργικά Συστήματα Ι, 2001

Τσιόκανος Ανδρέας, ΑΤΕΙ Θεσσαλίας, Συγκριτική Δοκιμή Λειτουργικού Συστήματος Windows Server 2008 με Server Ανοιχτού Κώδικα (Ubuntu Server) σε επίπεδο εκμετάλλευσης συγκεκριμένων πόρων υλικού & Web υπηρεσιών, 2014

Χαρίδημος Βέργος, Νικόλαος Κωστάρας, Εγχειρίδιο Χρήσης ΑΤ91, Πανεπιστήμιο Πατρών, 2007

Ξένη Βιβλιογραφία

Alan A. A. Donovan & Brian W. Kernighan, The Go Programming Language, 2016

Alexandrescu Andrei, The D Programming Language, Pearson Education, 2010

Bitterling Patrick, Operating System Kernels, University of Berlin, Germany, 2009

Chen Haogang, Yandong Mao, Xi Wang, Dong Zhou, Nickolai Zeldovich, M. Frans Kaashoek, Mit Csail, Linux kernel vulnerabilities: State-of-the-art defenses and open problems, Tsinghua University

D’Cunha Nihal A., Massimiliano Pala, Sean W. Smith, Programming Languages Security: A Survey For Practitioners, Dartmouth College

Eck David J, Hobart and William Smith Colleges, Introduction to Programming Using Java, 1996–2014
Flanagan David, JavaScript: The Definitive Guide, Sixth Edition, O’Reilly Media, 2011

Kernighan Brian W., Dennis M. Ritchie. "C programming language", AT&T Bell Laboratories, 2nd edition, 1988.

McCarthy John, Reminiscences on the history of time sharing, Stanford University, 1983

Normark Kurt, Object-oriented Programming in C#, Aalborg University, Denmark 2010

Sampson Geoffrey & Ventus Publishing ApS, Perl for Beginners, 2010

Silberschatz Abraham, Peter Baer Galvin, Greg Gagne, "Operating System Concepts", John Wiley and Sons Inc. 8th edition, 2009.

Stallings William, "Operating Systems: Internals and Design Principles", Prentice Hall, 7th edition, 2011.

Stallings William,The unix operating system, Fifth Edition, 2005

Stanley Phillip, Inferno Programming With Limbo, Marbell Carnegie Mellon University, 2003

Stroustrup Bjarne (creator of the c++), The C++ programming language, fourth edition, 2013

Tannenbaum Andrew S, "Modern Operating Systems", Prentice Hall, 5th edition, 2009

Torvalds Linus, Linux: a Portable Operating System, University Of Helsinki, Department of Computer Science, 1997

Walden David, Tom Van Vleck, The Compatible Time Sharing System (1961–1973), Fiftieth Anniversary Commemorative Overview, IEEE Computer Society, 2001

Wheeler, D. J.,The use of sub-routines in programmes, Cambridge University 1952

Διαδικτυακοί σύνδεσμοι

Αραποστάθης Μ. (2011), Λειτουργικό Σύστημα Unix – Linux, Διαθέσιμο στο διαδικτυακό τόπο: http://eclass.sch.gr/modules/document/file.php/T178119/%CE%A3%CE%97%CE%9C%CE%95%CE%99%CE%A9%CE%A3%CE%95%CE%99%CE%A3.pdf (τελευταία πρόσβαση 26/03/2016)

C (γλώσσα προγραμματισμού) 2016, Διαθέσιμο στο διαδικτυακό τόπο: https://el.wikipedia.org/wiki/C_(%CE%B3%CE%BB%CF%8E%CF%83%CF%83%CE%B1_%CF%80%CF%81%CE%BF%CE%B3%CF%81%CE%B1%CE%BC%CE%BC%CE%B1%CF%84%CE%B9%CF%83%CE%BC%CE%BF%CF%8D) (τελευταία πρόσβαση 30/03/2016)

Λειτουργικό σύστημα 2016, Διαθέσιμο στο διαδικτυακό τόπο: https://el.wikipedia.org/wiki/%CE%9B%CE%B5%CE%B9%CF%84%CE%BF%CF%85%CF%81%CE%B3%CE%B9%CE%BA%CF%8C_%CF%83%CF%8D%CF%83%CF%84%CE%B7%CE%BC%CE%B1 (τελευταία πρόσβαση 30/03/2016)

Παρουσίαση του FreeBSD (2013), Διαθέσιμο στο διαδικτυακό τόπο: http://www.freebsdworld.gr/diktia/FreeBSD-Intro.pdf (τελευταία πρόσβαση 30/03/2016)

Το λειτουργικό σύστημα του Solaris (2014), Διαθέσιμο στο διαδικτυακό τόπο: http://osarena.net/distros-roms/to-litourgiko-sistima-solaris.html (τελευταία πρόσβαση 29/03/2016)

Τι είναι Unix, BSD, Linux και τι UbuntuBSD (2016), Διαθέσιμο στο διαδικτυακό τόπο: https://iguru.gr/2016/04/03/57664/unix-bsd-linux-ubuntubsd/ (τελευταία πρόσβαση 04/04/2016)

A Brief History of User Interfaces, Available at http://www.catb.org/esr/writings/taouu/html/ch02.html (last access 16/03/2016)

Dillon Matthew, Virtual Memory System, Available at https://www.freebsd.org/doc/en/books/arch-handbook/vm.html, (last access 24/03/2016)

IBM AIX (2015), Available at http://www-03.ibm.com/systems/power/software/aix/ (last access 30/03/2016)

Inside the Unix Kernel, Available at http://www.pearsonhighered.com/samplechapter/0130187062.pdf (last access 18/03/16)

Linux vs. Unix, Available at http://www.diffen.com/difference/Linux_vs_Unix (last access 27/03/2016)

LSM Framework (2003) http://www.hep.by/gnu/kernel/lsm/framework.html (last access 13/04/2016)

MCC Interim Linux – Read me files, Available at http://www.ibiblio.org/pub/historic-linux/distributions/MCC-1.0/1.0/documentation/README (last access 01/04/2016)

McJones Paul (2015), History of ALGOL, Available at http://www.softwarepreservation.org/projects/ALGOL, (last access 01/03/2016)

OpenIndiana (2016), Available at http://distrowatch.com/table.php?distribution=openindiana (last access 30/03/2016)

Top 50 Products By Total Number Of "Distinct" Vulnerabilities in 2015, Available at https://www.cvedetails.com/top-50-products.php?year=2015 (last access 30/03/2016)

Volkerding Patrick (2013), The slackware linux project, Available at http://www.slackware.com/announce/14.1.php, (last access 20/02/2016)

Vivek Gite (2006), Difference Between Linux and UNIX, Available at http://www.cyberciti.biz/faq/what-is-the-difference-between-linux-and-unix/ (last access 27/03/2016)

Unix timeline (2015), Available at https://commons.wikimedia.org/wiki/File:Unix_timeline.en.svg (last access 27/03/2016)

UnixWare 7.1.4 (2015), Available at http://www.sco.com/products/unixware714/ (last access 29/03/2016)

GNU/Linux Distribution Timeline (2013), Available at http://futurist.se/gldt/ (last access 02/04/2016)