ΛΕΙΤΟΥΡΓΚΑ ΣΥΣΤΗΜΑΤΑ

Ο.Π.Α

4ο ΕΞΑΜΗΝΟ

ΤΖΕΝΗ ΜΠΟΛΕΝΑ

Περιεχόμενα

ΕΝΟΤΗΤΑ 1: ΕΙΣΑΓΩΓΗ

Τι είναι το λειτουργικό σύστημα;

Είναι ένας διαχειριστής πόρων(επεξεργαστής, μνήμη, δίσκος…), δλδ μπορεί να χρονοπρογραμματίζει τον επεξεργαστή ή να δίνει πρόσβαση στην μνήμη σε ένα πρόγραμμα. Επίσης παρέχει διεπαφή προ τον προγραμματιστή, ο οποίος δεν επικοινωνεί απευθείας με το υλικό.

Καταστάσεις λειτουργίας υπολογιστή:

- Κατάσταση πυρήνα ή επόπτη: εκτελείτε το ΛΣ οπότε μπορεί να εκτελεστεί οποιαδήποτε εντολή της μηχανής
- Κατάσταση χρήστη: διεπαφή του χρήστη και εκτέλεση εφαρμογών. Δεν επιτρέπεται η εκτέλεση ορισμένων εντολών όπως να γράψει ο χρήστης απευθείας στον δίσκο

Τι περιλαμβάνει το ΛΣ;

- Βασικό λογισμικό που εκτελείται σε κατάσταση πυρήνα
- Πιθανό και προνομοιούχο λογισμικό επιπέδου χρήστη που να έχει σχέση με το ΛΣ

Η **επεκτεταμένη μηχανή** που φτιάχνει το ΛΣ είναι διεπαφή για τον προγραμματιστή. Το υλικό μπορεί να διαφέρει από μηχανή σε μηχανή, οπότε το ΛΣ το παρουσιάζει με τρόπο όμοιο στον προγραμματιστή για να μην επηρεάζεται από το υλικό, παρέχει λογικές αφαιρέσεις. (Όψη από πάνω)

Το ΛΣ ως διαχειρηστής πόρων(όψη από κάτω): Πολλά προγράμματα εκτελούνται ταυτόχρονα και μοιράζονται τους πόρους, αν και δίνεται η αίσθηση ότι εκτελείται το καθένα σε διαφορετική μηχανή.

Πολύπλεξη πόρων σε δύο άξονες:

- **Χρόνο: διαδοχική χρήση ΚΜΕ από διαφορετικά προγράμματα**
- **Χώρος: συνύπαρξη προγραμμάτων στην μνήμη**

Εκκίνηση ΛΣ: Όταν ανοίξουμε τον υπολογιστή ξεκινάει το πρόγραμμα εκκίνησης(**BIOS**) το οποίο βρίσκεται στη ROM ή σε Flash drive. Το πρόγραμμα αυτό θα δει ποιες συσκευές

είναι συνδεδεμένες και θα δώσει τον έλεγχο σε μία από αυτές για να γίνει η εκκίνηση του ΛΣ.

Είδη ΛΣ:

- **ΛΣ για μεγάλους υπολογιστές(mainframes**). Χρησιμοποιούνται σε τράπεζες και έχουν μεγάλες δυνατότητες I/O
- **ΛΣ για διακομιστές(servers).** Διαχείριση αιτημάτων πολλών χρηστών, π.χ webservers. Έχουν χαμηλό χρόνο απόκρισης.
- **ΛΣ για πολυεπεξεργαστές(multiprocessor).** Συνήθως παραλλαγές των ΛΣ για διακομιστές
- **ΛΣ για προσωπικούς υπολογιστές(PC).**
- **ΛΣ για υπολογιστές χειρός(handhelds).** Smartphones και tablets. Όμοια με ΛΣ PC αλλά πιο απλά
- **Ενσωματωμένα ΛΣ(embedded).** Σε DVD's…, εκτελούν προκαθορισμένα προγράμματα
- **ΛΣ κόμβων αισθητήρων(sensors).** Πολύ απλά συστήματα οδηγούμενα από γεγονότα
- **ΛΣ πραγματικού χρόνου.** Ειδική κατηγορία των embedded ή PC. Χωρίζονται σε δυο κατηγορίες: **1) Αυστηρά συστήματα πραγματικού χρόνου.** Όλες οι εργασίες πρέπει να εκτελούνται εγκαίρως αλλιώς θα υπάρξουν καταστροφικές συνέπειες. **2) Ήπια συστήματα πραγματικού χρόνου.** Αν σπάνια δεν έχουμε απόκριση τότε all good.
- **ΛΣ έξυπνων καρτών(smart cards).** Μπορούν να κάνουν κρυπτογράφηση, πολύ απλά.

ΕΝΝΟΙΕΣ ΛΣ

Διεργασίες: είναι προγράμματα που εκτελούνται από έναν χρήστη ή μια ομάδα χρηστών. Έχουν δικό τους χώρο διευθύνσεων που περιλαμβάνει το πρόγραμμα και τα δεδομένα καθώς και πόρους(καταχωρητές, ανοιχτά αρχεία, σήματα). Επειδή μια διεργασία μπορεί κάποια στιγμή να σταματήσει για κάποιο χρονικό διάστημα υπάρχει ο πίνακας διεργασιών στον οποίο αποθηκεύονται οι πόροι και βρίσκεται στην κύρια μνήμη. Ο χώρος διευθύνσεων μπορεί να βρίσκεται εν μέρει στην μνήμη. Το ΛΣ δίνει τη δυνατότητα στον προγραμματιστή να κάνει κλίσεις συστήματος ώστε να ξεκινήσει και να τερματίσει το πρόγραμμα. Αρκετές φορές μπορεί να υπάρξει και επικοινωνία μεταξύ των διεργασιών.

Χώροι διευθύνσεων: περιλαμβάνει το πρόγραμμα και τα δεδομένα. Το ΛΣ δίνει την αίσθηση ότι το κάθε πρόγραμμα τρέχει μόνο του, ωστόσο στην πραγματικότητα μοιράζεται την μνήμη με άλλα προγράμματα. Ένα πρόγραμμα το οποίο μπορεί να είναι

μεγαλύτερο από τη φυσική μνήμη μπορεί και τρέχει και δίνεται η αίσθηση ότι βρίσκεται όλο στην μνήμη χάρη στην εικονική μνήμη(μεγάλος χώρος διευθύνσεων). Ωστόσο δεν είναι όλο το πρόγραμμα στην μνήμη, μόνο το μέρος που επεξεργάζεται αυτή τη στιγμή, το υπόλοιπο βρίσκεται στον δίσκο ή σε κάποιον άλλο αποθηκευτικό χώρο. Η εικονική μνήμη επιτρέπει τη συνύπαρξη διεργασιών στην μνήμη.

Αρχεία: το ΛΣ πάνω από κάθε συσκευή αποθήκευσης δημιουργεί ένα λογικό σύστημα αρχείων. Οπότε ενώ στην πραγματικότητα τα δεδομένα αποθηκεύονται σε συνεχόμενα block μνήμης, αυτό που βλέπει ο χρήστης είναι ένα αρχείο με συγκεκριμένο όνομα δικής τους επιλογής και τοποθετημένο σε κατάλογο της επιλογής τους. Υπάρχει και ο περιγραφέας αρχείου ο οποίος δημιουργείται όταν ανοίγουμε ένα αρχείο και είναι ένας δείκτης πάνω στο ανοιχτό αρχείο. Στη συνέχεια αν θέλουμε μπορούμε να δείνουμε τον περιγραφέα αρχείου αντί το όνομα για να κάνουμε τις επόμενες λειτουργίες(ανάγνωση, εγγραφή..). Τα αρχεία οργανώνονται σε καταλόγους σε δεντρική δομή. Επίσης υπάρχουν και ειδικά αρχεία τα οποία όμως δεν είναι αρχεία αλλά κρύβουν πίσω τους συσκευές. Τέλος υπάρχουν οι αγωγοί ή σωληνώσεις που είναι ψευδοαρχεία και βρίσκονται ανάμεσα σε δύο διεργασίες και τους επιτρέπει να επικοινωνούν μεταξύ τους.

Κέλυφος(γραμμή εντολών, γραφική διεπαφή): δεν είναι μέρος του ΛΣ, επιτρέπει στον χρήστη να επικοινωνεί με το λειτουργικό.

ΚΛΗΣΕΙΣ ΣΥΣΤΗΜΑΤΟΣ

Το ΛΣ εκτός από διαχειριστής πόρων είναι και εκτεταμένη μηχανή. Η εκτεταμένη μηχανή που παρέχει το ΛΣ σε έναν προγραμματιστή είναι οι μη προνομιούχες εντολές που είναι διαθέσιμες στον προγραμματιστή καθώς και μια σειρά από έξτρα δυνατότητες που παρέχει το ΛΣ στον προγραμματιστή μέσω των κλήσεων συστήματος. Οι κλήσεις συστήματος είναι μια σειρά από εντολές που επιτρέπει στον προγραμματιστή να κάνει λειτουργίες που το ΛΣ του κρύβει. Είναι και η βασική διεπαφή του προγραμματιστή με το ΛΣ.

ΔΟΜΗ ΛΣ(δλδ από τι αποτελείται, πώς οργανώνεται)

ΜΟΝΟΛΗΘΙΚΑ ΣΥΣΤΗΜΑΤΑ: είναι ένα ενιαίο πρόγραμμα, οργανωμένο και έχει διαδικασίες που κάνουν διαφορές λειτουργίες, αλλά επειδή η κάθε διαδικασία είναι ορατή σε όλες τις υπόλοιπες δεν είναι προστατευμένο και ευκολά μπορούν να γίνουν λάθη που θα προκαλέσουν ζημίες. Υπάρχει και μεγάλη δυσκολία στην συντήρηση τους καθώς και στην διόρθωση σφαλμάτων.

ΠΟΛΥΕΠΙΠΕΔΑ ΣΥΣΤΗΜΑΤΑ: πολλά επίπεδα απομονωμένα, όπου το καθένα κάνει/παρέχει χωριστές λειτουργίες. Κάθε επίπεδο παρέχει υπηρεσίες στα παραπάνω.

ΜΙΚΡΟΠΥΡΗΝΕΣ: απλούστευση των πολυεπίπεδων συστημάτων. Στην ουσία υπάρχουν δύο επίπεδα, το επίπεδο του μικροπυρήνα όπου εκτελούνται ελάχιστες λειτουργίες (χρονοπρογραμματισμός, διεργασίες…) και το επίπεδο που έχει όλα τα υπόλοιπα, δλδ λειτουργίες που μπορούν να γίνουν σε επίπεδο χρήστη.

ΜΟΝΤΕΛΟ ΠΕΛΑΤΗ-ΕΞΥΠΗΡΕΤΗΤΗ: υπάρχουν πολλοί πυρήνες που θα κάνουν διαφορετική λειτουργία. Ο πελάτης που έχει δικό του πυρήνα όταν θέλει να κάνει κάτι στέλνει μήνυμα σε έναν πυρήνα και περιμένει απάντηση. Σκοπός είναι να σπάσουμε το ΛΣ σε μικρότερα κομμάτια.

ΕΙΚΟΝΙΚΕΣ ΜΗΧΑΝΕΣ: πολλά αυτόνομα λειτουργικά συστήματα τρέχουν πάνω από το ίδιο υλικό, έχουμε πολλά υπολειτουργικά πάνω στο ίδιο λειτουργικό.

ΕΝΟΤΗΤΑ 2: ΔΙΕΡΓΑΣΙΕΣ ΚΑΙ ΝΗΜΑΤΑ

ΔΙΕΡΓΑΣΙΕΣ

Η **διεργασία** είναι ένα εκτελούμενο πρόγραμμα που εκτός από τον κώδικα αποτελείται από δεδομένα, καταχωρητές και έναν μετρητή προγράμματος. Σε κάθε ΛΣ υπάρχουν πολλές διεργασίες οι οποίες εκτελούνται ψευδοταυτόχρονα με γρήγορες εναλλαγές μεταξύ τους. Οι εναλλαγή αυτή των διεργασιών ονομάζεται πολυπρογραμματισμός. Η ΚΜΕ απασχολείται με μία μόνο διεργασία τη φορά. Ο χρόνος εκτέλεσης των διεργασιών είναι απρόβλεπτος. Επίσης πρέπει να γίνει διάκριση μεταξύ του προγράμματος και της διεργασίας. Το πρόγραμμα είναι η συνταγή και είναι στατικό, ενώ η διεργασία είναι η εκτέλεση του προγράμματος και μεταβάλετε κάθε φορά που εκτελείται το πρόγραμμα.

Δημιουργία διεργασιών: σε κλειστά συστήματα(embedded, real time) δημιουργούνται όλες μαζί(στην αρχή), αλλά γενικά δημιουργούνται δυναμικά.

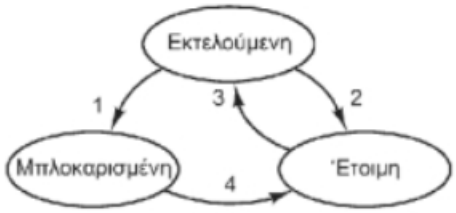
Τρόποι δημιουργίας διεργασίας:

- Αρχικοποίηση του συστήματος, όπως η διεργασία σύνδεσης ή ο έλεγχος για εισερχόμενα email
- Δημιουργία διεργασίας από άλλη διεργασία για καταμερισμό εργασίας
- Αίτηση χρήστη για δημιουργία διεργασίας
- Εκκίνηση διεργασίας δέσμης, δλδ ξεκινάει όταν έρθει η σειρά της, μπορεί επίσης να έχει οριστεί να ξεκινήσει μια συγκεκριμένη ώρα.

Τρόποι τερματισμού διεργασίας:

- Κανονική έξοδος
- Έξοδος που προκλήθηκε από σφάλμα(εκούσια, π.χ λάθος παράμετροι)
- Μοιραίο σφάλμα(ακούσια, π.χ διαίρεση με το 0)
- Θανάτωση από άλλη διεργασία(ακούσια)

Καταστάσεις διεργασιών



ΝΗΜΑΤΑ

Χρήση των νημάτων

Τα νήματα είναι ένα υποσύνολο μιας διεργασίας. Μοιράζονται την μνήμη της ίδιας διεργασίας και επικοινωνούν μέσω της κοινής μνήμης, αλλά έχουν ξεχωριστούς καταχωρητές για να γνωρίζουμε την κατάσταση που βρίσκονται. Η δημιουργία τους είναι πιο οικονομική από αυτής των διεργασιών(πρέπει για να δημιουργηθούν από κατάσταση χρήστη να πάνε σε πυρήνα και ξανά σε χρήστη).

Παράδειγμα χρησιμοποίησης νημάτων: ένας επεξεργαστής κειμένου παρακολουθεί τι πληκτρολογεί ο χρήστης, το εμφανίζει στην οθόνη, κάνει ορθογραφικό έλεγχο και περιοδικά γράφει τα δεδομένα στον δίσκο. Η διεργασία αυτή αποτελείται από τρία νήματα, το πληκτρολόγιο, τον ορθογραφικό ελεγκτή και την εγγραφή στον δίσκο τα οποία ενεργούν πάνω σε κοινή μνήμη και συγκεκριμένα πάνω στο πρόγραμμα και στα δεδομένα της διεργασίας που είναι το κείμενο που γράφουμε.

Η κύρια χρήση των νημάτων είναι σε εξυπηρετητές, εκεί ένα αίτημα δίνεται σε ένα νήμα, και έχουμε πολλά αιτήματα να εξυπηρετούνται ταυτόχρονα. Επίσης αν ένα νήμα σταματήσει, επειδή π.χ περιμένει ένα δεδομένο εισόδου, τότε δεν μπλοκάρεται η διεργασία αφού υπάρχουν πολλά άλλα νήματα που εξυπηρετούν και είναι ενεργά.

Κλασικό μοντέλο νημάτων

- Η διεργασία ομαδοποιεί ένα σύνολο πόρων
  - Κώδικας, δεδομένα, αρχεία, σήματα
- Κάθε νήμα είναι μία ροή εκτέλεσης μέσα στη διεργασία
  - Μετρητής, καταχωρητές, στοίβα
- Τα νήματα μοιράζονται τους πόρους της ίδιας διεργασίας

Τα νήματα δεν απομονόνονται όπως οι διεργασίες, οι πόροι είναι κοινοί και μπορούν να ανταλλάσσουν δεδομένα μεταξύ τους και να καταστρέφουν το ένα το άλλο.

Καταστάσεις νημάτων: εκτελούμενο, έτοιμο, μπλοκαρισμένο.

Δημιουργία και καταστροφή νημάτων: το βασικό νήμα τις διεργασίας μπορεί να δημιουργήσει και άλλα νήματα. Η καταστροφή είναι παρόμοια με αυτή των διεργασιών. Επίσης ένα νήμα μπορεί να παραδώσει την ΚΜΕ οικειοθελώς όταν π.χ τελειώσει την εργασία που έκανε ώστε να μην καταστρέφουμε συνέχεια νήματα.

ΥΛΟΠΟΙΗΣΗ ΝΗΜΑΤΩΝ

- **Σε επίπεδο χρήστη:** η διεργασία υλοποιεί τα νήματα. Το ΛΣ δεν γνωρίζει τίποτα για τα νήματα. Υπάρχει μια βιβλιοθήκη που δημιουργεί τα νήματα. Ωστόσο επειδή το ΛΣ δεν γνωρίζει την ύπαρξη νημάτων όταν μπλοκαριστεί ένα νήμα μπλοκάρεται και όλη η διεργασία, αυτό μπορούμε να το αποφύγουμε με ασύγχρονες κλίσεις αλλά δεν είναι το καλύτερο. Επίσης να νήματα δεν μπορούν να διακοπούν από το σύστημα επειδή αυτό δεν γνωρίζει την ύπαρξη τους και θα πρέπει να παραδοθούν οικειοθελώς.
- **Σε επίπεδο πυρήνα:** το ΛΣ ασχολείται με την υλοποίηση των νημάτων. Όλοι οι πίνακες και κλήσεις εμπλέκουν τον πυρήνα. Το μπλοκάρισμα αντιμετωπίζεται από τον πυρήνα επιλέγοντας άλλο νήμα τις διεργασίας. Επειδή το κόστος δημιουργίας νέων νημάτων είναι μεγάλο δημιουργούνται από την αρχή αρκετά νήματα τα οποία βρίσκονται σε αδράνεια.
- **Υβριδικές υλοποιήσεις:** δημιουργούνται τα νήματα στον πυρήνα και πάνω σε αυτά δημιουργούνται πολλά άλλα επιπέδου χρήστη.
- **Αναδυόμενα:** δημιουργείται νέο νήματα όταν έρχεται νέο αίτημα.

ΔΙΑΔΙΕΡΓΑΣΙΑΚΗ ΕΠΙΚΟΙΝΩΝΙΑ

Οι διεργασίες είναι απομονωμένες και επικοινωνούν με αυστηρά ελεγχόμενο τρόπο. Υπάρχουν τρείς λόγοι που μπορεί να χρειαστεί να επικοινωνήσουν οι διεργασίες μεταξύ τους:

- Μεταβίβαση πληροφοριών μεταξύ τους
- Αποφυγή συγκρούσεων για κοινούς πόρους
- Εξαρτήσεις ανάμεσα στις διεργασίες

Αμοιβαίος αποκλεισμός: βασική τεχνική για αποφυγή συνθηκών ανταγωνισμού.

ΛΥΣΕΙΣ ΣΤΟ ΠΡΟΒΛΗΜΑ ΤΟΥ ΑΜΟΙΒΑΙΟΥ ΑΠΟΚΛΙΣΜΟΥ(ΔΛΔ ΣΤΗΝ ΥΛΟΠΟΙΗΣΗ ΤΩΝ ΚΡΙΣΙΜΩΝ ΠΕΡΙΟΧΩΝ)

ΑΝΑΜΟΝΗ ΜΕ ΑΠΑΣΧΟΛΗΣΗ(BUSY WAITING): στην αναμονή με απασχόληση μια μόνο διεργασία είναι μέσα στην κρίσιμη περιοχή και οι άλλες ενώ περιμένουν να μπορούν προσπαθούν συνέχεια να δουν αν μπορούν να χρησιμοποιήσουν το κρίσιμο πόρο σπαταλώντας έτσι πολλούς άλλους πόρους.

Τρόποι για να πετύχουμε το busy waiting:

- **Απενεργοποίηση των διακοπών:** όταν μια διεργασία λάβει τον κρίσιμο πόρο τότε απενεργοποίησε όλες τις διακοπές του έτσι ώστε να μην μπορεί το σύστημα να το διακόψει, αλλά να διακοπή μόνο με δική του θέληση το οποίο προφανώς δεν θα το κάνει αφού είναι μέσα στην κρίσιμη περιοχή. Όταν φύγει από την κρίσιμη περιοχή θα ενεργοποίηση ξανά τις διακοπές του. Η απενεργοποίηση των διακοπών ωστόσο δεν είναι η βέλτιστη λύση αφού αν προκληθεί σφάλμα στην διεργασία τότε μπορεί να κολλήσει όλο το σύστημα.
- **Μεταβλητές κλειδώματος:** υπάρχει μια μεταβλητή για κάθε κρίσιμο σημείο(πόρο). Αν ο πόρος δεν χρησιμοποιείται η μεταβλητή έχει τιμή 0. Μια διεργασία που θα διαβάσει τιμή 0 θα πάει να την κάνει 1 για να στην κρίσιμη περιοχή, αλλά αν δύο διεργασίες διαβάσουν ταυτόχρονα τιμή 0 τότε θα πάνε και οι δύο να μπουν στην κρίσιμη περιοχή, άρα η λύση αυτή δεν δουλεύει.
- **Αυστηρή εναλλαγή:** χρήση μεταβλητής που δείχνει ποιος έχει σειρά. Μια διεργασία θα μπει στην κρίσιμη περιοχή μόνο όταν έρθει η σειρά της και όταν βγει από την κρίσιμη περιοχή θα γράψει πότε είναι η σειρά της ξανά. Το πρόβλημα αυτό είναι ότι παραβιάζονται κάποιες βασικές συνθήκες για όταν θέλουμε να μπούμε στην κρίσιμη περιοχή, Μια διεργασία που θέλει τώρα να μπει στην κρίσιμη περιοχή θα χρειαστεί να περιμένει να μπει πρώτα η διεργασία που έχει σειρά η οποία αυτή τη στιγμή μπορεί να βρίσκεται εκτός κρίσιμης περιοχής.
- **Η λύση του Peterson:** Υπάρχει ένας πίνακας Boolean όπου κάθε διεργασία γράφει αν θέλει να μπει στην κρίσιμη περιοχή ή όχι. Και υπάρχει και μια μεταβλητή turn που πάει η διεργασία να γράψει ότι είναι η σειρά της. Η τρέχουσα διεργασία μπαίνει σε loop και παραμένει εκεί όσο είναι η σειρά της αλλά ενδιαφέρεται κάποια άλλη. Αν δεν είναι η σειρά μας ή είναι η σειρά μας και δεν ενδιαφέρεται κάποια άλλη τελειώνει το loop και μπαίνουμε στην κρίσιμη περιοχή. Φεύγοντας δηλώνουμε ότι δεν ενδιαφερόμαστε πλέον για τον πόρο.
- **Η εντολή TSL(Test And Set Lock):** η εντολή αυτή παίρνει δύο παραμέτρους τον TSL register και το lock που είναι μια θέση μνήμης. Αντιγράφεται η τιμή που περιέχει η μεταβλητή κλειδώματος lock στον register και αποθηκεύει στην θέση lock την τιμή 1. Εκτελείται η TSL από μια διεργασία, αν η τιμή του register ήταν 0 τότε σημαίνει ότι μπήκε στην κρίσιμη περιοχή και έχει γραφτεί στον register και στο lock τιμή 1. Όταν βγει από την κρίσιμη περιοχή με μια εντολή move θα κάνει

τον register και το lock σε τιμή 0. Αν όμως ο register είχε τιμή 1 τότε κάποια άλλη διεργασία είναι στον κρίσιμο πόρο και μπαίνει σε ένα loop και ξανατρέχει την εντολή TSL.

- **Η εντολή XCHG:** παραλλαγή της TSL. Κάνει τον register 1 και στην συνέχεια ανταλλάσσει τις τιμές των register και lock, αν ο register είναι 1 σημαίνει ότι άλλη διεργασία είναι στην κρίσιμη περιοχή και ξανά-προσπαθεί. Αν η τιμή του register είναι 0 μπαίνει και όταν βγαίνει και κάνει την τιμή του lock 0. Έχουμε ξανά δύο βήματα σε μια εντολή οπότε δεν υπάρχει ο κίνδυνος να διαβάσουν δυο διεργασίες ταυτόχρονα τιμή 0 και να μπουν στην κρίσιμη περιοχή.

ΛΗΘΑΡΓΟΣ ΚΑΙ ΑΦΥΠΝΗΣΗ(κλήσεις sleep και wake up): όταν μια διεργασία θέλει να μπει στην κρίσιμη περιοχή αλλά βλέπει ότι είναι μια άλλη μπαίνει σε λήθαργο και περιμένει να την ξυπνήσει το σύστημα. Το wake up γίνεται όταν μια διεργασία βγαίνει από την κρίσιμη περιοχή και ενημερώνει το σύστημα για να ξυπνήσει μια άλλη.

Πρόβλημα παραγωγού-καταναλωτή: έστω ότι έχουμε Ν θέσεις. Αν έχουν γεμίσει Ν θέσεις ο παραγωγός κοιμάται, στις Ν-1 γεμάτες ξυπνά τον καταναλωτή. Ο καταναλωτής στις Ν κενές θέσεις μπλοκάρεται και στις Ν-1 κενές ξυπνά τον παραγωγό.

ΣΗΜΑΤΟΦΟΡΟΙ: οι σηματοφόροι παίρνουν θετικές τιμές ή 0. Ορίζονται δύο πράξεις πάνω σε σηματοφόρους, η **up** και **down**. Η down αν δει ότι ο σηματοφόρος έχει τιμή>0 τότε τον μειώνει κατά 1 και συνεχίζεται η διεργασία, αλλιώς δλδ αν είναι 0, μπλοκάρεται η διεργασία. Η up όταν τελειώσει μια διεργασία αν έχουν μπλοκαριστεί διεργασίες αφυπνίζεται μια αλλιώς αυξάνεται κατά 1. Η up και down πρέπει να γίνονται σε ένα βήμα, δλδ με αμοιβαίο αποκλεισμό και γίνονται με χρήση της εντολής TSL ή XCHG.

MUTEX: απλή μορφή δυαδικού σηματοφορέα. Υπάρχουν δύο λειτουργίες, mutex_lock και mutex_unlock. Αν το Mutex είναι locked καλούμε τον χρονοπρογραμματιστή ο οποίος μας πάει για ύπνο και αφού μας αφυπνίσει ξαναδοκιμάζουμε. Αν είναι unlocked κάνουμε lock. Το mutex υλοποιείται στον χώρο του χρήστη και τα νήματα μπλοκάρονται προσωρινά, δεν σπαταλάμε πολλούς πόρους σε αντίθεση με το busy waiting. Τα mutex έχουν νόημα σε συνεργατικό περιβάλλον όπως στα νήματα αφού σίγουρα ένα από αυτά θα μπει στο mutex. Τα mutex δεν επαρκούν για πιο σύνθετες συνθήκες και για αυτόν τον λόγο έχουμε τις **μεταβλητές συνθήκης.** Pthread_cond_wait() περιμένει μέχρι να συμβεί κάτι**.** Το wait βρίσκεται μέσα σε mutex.

ΕΛΕΓΚΤΕΣ: είναι μια δομή της γλώσσας προγραμματισμού που προσπαθεί να επιβάλλει τον αμοιβαίο αποκλεισμό. Αποτελείται από δεδομένα και διαδικασίες(σαν αντικείμενο) και μέσα του μπορεί να είναι ενεργή μια μόνο διαδικασία. Ένας ελεγκτής μπορεί να έχει και signal και wait.

ΜΕΤΑΒΙΒΑΣΗ ΜΗΝΥΜΑΤΩΝ: είναι γενική τεχνική και η μόνη κατάλληλη για κατανεμημένα συστήματα. Υπάρχουν δυο κλήσεις, η send(dest, &msg) και receive(src, &msg).

ΦΡΑΓΜΑΤΑ: οι διεργασίες εκτελούνται σε φάσεις και περιμένουμε να φτάσουν όλες μέχρι ένα συγκεκριμένο σημείο(barrier), άρα όσες έφτασαν πιο νωρίς μπλοκάρονται και μετά ξεμπλοκάρονται και συνεχίζουν. Επιτρέπει συγχρονισμό παράλληλων υπολογισμών.

ΧΡΟΝΟΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

Ο χρονοπρογραμματισμός αναφέρεται στο ποια διεργασία θα εκτελεστεί μετά από κάποια διακοπή. Ο χρονοπρογραμματισμός γίνεται:

- Όταν δημιουργείται/τερματίζεται μια διεργασία
- Όταν μια διεργασία μπλοκάρεται για κάποιο λόγο
- Όταν προκύπτει μια διακοπή εισόδου/εξόδου
- Όταν προκύπτει διακοπή χρονομέτρου

Επειδή οι επεξεργαστές βελτιώνονται πιο γρήγορα από τους δίσκους οι περισσότερες διεργασίες εξαρτώνται από την είσοδο/έξοδο.

Υπάρχουν **προεκτοπιστικοί αλγόριθμοι** όπου η διεργασία εκτελείται μέχρι κάποιο όριο και **μη προεκτοπιστικοί αλγόριθμοι** όπου η διεργασία εκτελείται μέχρι να μπλοκαριστεί.

ΚΑΤΗΓΟΡΙΕΣ ΑΛΓΟΡΙΘΜΩΝ ΧΡΟΝΟΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

- **Συστήματα δέσμης:** μεγάλος φόρτος εργασίας, μας ενδιαφέρει η αύξηση της επίδοσης. Μη προεκτοπιστικοί ή προεκτοπιστικοί με μεγάλο διάστημα. Στόχος η μεγάλη διεκπεραιωτική ικανότητα και μικρός μέσος χρόνος. Υψηλή αξιοποίηση επεξεργαστή.
- **Συστήματα αλληλεπίδρασης(PC):** προεκτοπιστικοί, όλες οι διεργασίες πρέπει να εκτελούνται από λίγο. Στόχος ο μικρός χρόνος απόκρισης.
- **Συστήματα πραγματικού χρόνου:** δεν είναι απαραίτητη η προεκτόπιση γιατί έχουν φτιαχτεί έτσι ώστε μετά από κάποιο διάστημα να εναλλάσσονται οι διεργασίες μεταξύ τους. Στόχος η τήρηση των deadlines των διεργασιών.

Συστήματα δέσμης

- Εξυπηρέτηση με τη σειρά άφιξης. Οι νέες διεργασίες μπαίνουν στο τέλος της ουράς.
- Εξυπηρέτηση με βάση τη μικρότερη διάρκεια.
- Εξυπηρέτηση με βάση το μικρότερο υπόλοιπο.

Συστήματα αλληλεπίδρασης

- **Προγραμματισμός εκ περιτροπής(round robin):** οι διεργασίες βρίσκονται σε μια ουρά και μια διεργασία εκτελείται μέχρι να λήξει ο χρόνος που έχει εξαρχής αποφασιστεί ή μέχρι να μπλοκαριστεί(ότι συμβεί πρώτο). Αν λήξει ο χρόνος πάει στο τέλος της ουράς. Πρέπει να καθορίσουμε το διάστημα εκτέλεσης(κβάντου), το οποίο πρέπει να είναι αρκετά μεγαλύτερο από το χρόνο εναλλαγής διεργασιών και αρκετά μικρό ώστε να έχουμε χαμηλό χρόνο απόκρισης.
- **Χρονοπρογραμματισμός με βάση την προτεραιότητα: ομαδοποίηση σε κλάσεις προτεραιοτήτων(**εκτέλεση εκ περιτροπής μέσα σε κάθε κλάση**). Ανάθεση προτεραιοτήτων:** Στατική ανάλογα με τη σημασία της διεργασίας και δυναμική ανάλογα με τον τρόπο εκτέλεσης.
- **Συστήματα πολλαπλών ουρών:** οι σταθερές προτεραιότητες έχουν κίνδυνο υποσιτισμού. Λύσεις για να μην συμβεί αυτό είναι οι εξής: Α)σταδιακή ελάττωση προτεραιότητας εκτελούμενης. Β)ανάθεση μεγαλύτερου κβάντου σε μικρές προτεραιότητες. Όταν μια διεργασία εξαντλεί το κβάντο υποβιβάζεται. Γ)ανάθεση προτεραιότητας με βάση τη συμπεριφορά, δλδ υψηλότερη προτεραιότητα αν χρησιμοποιεί τερματικό, χαμηλότερη αν εξαντλεί όλο το κβάντο.
- **Εξυπηρέτηση με βάση τη μικρότερη διάρκεια:** θεωρούμε κάθε νέο διάστημα εκτέλεσης ως νέα εργασία και βλέποντας τι έχει κάνει στο παρελθόν προσπαθούμε να μαντέψουμε για πόσο θα εκτελεστεί.
- **Εγγυημένος χρονοπρογραμματισμός:** αν έχουμε n χρήστες τότε ο καθένας παίρνει 1/n της ΚΜΕ. Υπολογίζεται ο λόγος χρόνου εκτέλεσης προς αναλογούντα και εκτελείται αυτή με το μικρότερο.
- **Χρονοπρογραμματισμός με λοταρία:** κάθε διεργασία παίρνει λαχνούς, όταν διακοπεί μια διεργασία αυτή που έχει τον λαχνό που επιλέχθηκε γίνεται εκτελούμενη. Στις ποιο σημαντικές διεργασίες δίνουμε περισσότερους λαχνούς.
- **Χρονοπρογραμματισμός δίκαιης κατανομής:** οι πόροι ανατίθενται δίκαια ανά χρήστη(όχι ανά διεργασία).

Συστήματα πραγματικού χρόνου

Οι διεργασίες θα πρέπει να εκτελούνται μέσα σε προθεσμίες. Έχουμε δυο κατηγορίες αναλόγως το τρόπο που εμφανίζονται τα γεγονότα:

- **Περιοδικά:** τα ίδια γεγονότα εμφανίζονται περιοδικά.
- **Απεριοδικά:** τα γεγονότα προκύπτουν απρόβλεπτα.

Έχουμε στατικούς και δυναμικούς αλγορίθμους.

ΧΡΟΝΟΠΡΟΓΡΑΜΜΤΙΣΜΟΣ ΝΗΜΑΤΩΝ

- **Σε επίπεδο χρήστη:** με βάση το κβάντο, αποφασίζει η διεργασία με ποια σειρά θα εκτελεστούν.
- **Σε επίπεδο πυρήνα:** μπορεί να αγνοούνται οι διεργασίες και να γίνει ο χρονοπρογραμματισμός σε επίπεδο νημάτων. Η εναλλαγή έχει κόστος και προτιμάται η εναλλαγή μεταξύ νημάτων ίδιας διεργασίας.

ΚΛΑΣΙΚΑ ΠΡΟΒΛΗΜΑΤΑ

- **ΤΟ ΔΕΙΠΝΟ ΤΩΝ ΦΙΛΟΣΟΦΩΝ**
  Πρόβλημα: έχουμε n πιάτα και n πιρούνια σε ένα τραπέζι και για να φάει κάποιος χρειάζεται και τα δυο γειτονικά πιρούνια. Η λύση να πάρουν όλοι το αριστερό πιρούνι και μετά το δεξί μπορεί να οδηγήσει σε αδιέξοδο διότι αν πάρουν όλοι το αριστερό πιρούνι ταυτόχρονα και κάνουν αν το δεξί δεν είναι διαθέσιμο τότε δεν θα φάνε ποτέ, deadlock. Η λύση είναι η εξής: έχουμε n σηματοφόρους για τους n φιλοσόφους. Όταν θέλει ένας φιλόσοφος να φάει και βλέπει αν μπορεί να πάρει το αριστερό και το δεξί πιρούνι αν ναι τότε τα παίρνει και κάνει down στον σηματοφόρο που δηλώνει ότι τρώει και όταν τελειώσει κάνει up, αν όχι τότε κάνει down στον σηματοφόρο που δηλώνει ότι είναι μπλοκαρισμένος.

- **ΠΡΟΒΛΗΜΑ ΑΝΑΓΝΩΣΤΩΝ-ΓΡΑΦΕΩΝ**
  Πρόβλημα: μοντελοποιεί πρόσβαση σε βάση δεδομένων. Μπορούν πολλοί να διαβάσουν την βάση ταυτόχρονα, αλλά μόνο ένας να γράφει. Η ερώτηση είναι «τι θα γίνει αν κάποιος θέλει να γράψει στη βάση; Θα συνεχίζουν να έρχονται αναγνώστες επειδή υπάρχουν ήδη άλλοι ή θα παραμείνουν μόνο οι αναγνώστες που ήδη διαβάζουν και οι υπόλοιποι θα περιμένουν να γράψει πρώτα ο γραφέας;»
  Λύση με σηματοφόρους. Έχουμε έναν σηματοφόρο **mutex** που χρησιμοποιείται για να κρατάμε το πλήθος των αναγνωστών στη βάση και έναν **db** για να ξέρουμε αν η βάση είναι ξεκλείδωτη(δλδ δεν την χρησιμοποιεί κανείς) ή όχι. Ο db επιτρέπει έναν γραφέα ή πολλούς αναγνώστες. Αν το db είναι up τότε κάνει down και έτσι δεν θα μπορούν να μπουν αναγνώστες. Όταν κάνει up μπαίνει ο πρώτος αναγνώστης και στη συνέχεια οι υπόλοιποι. Η λύση αυτή δίνει προτεραιότητα στον γραφέα.

ΕΝΟΤΗΤΑ 3: ΔΙΑΧΕΙΡΙΣΗ ΜΝΗΜΗΣ

Στο ΛΣ έχουμε τον **διαχειριστή μνήμης** ο οποίος:

- Γνωρίζει ποια τμήματα της μνήμης χρησιμοποιούνται
- Ασχολείται με πόση μνήμη παραχωρείται σε κάθε διεργασία
- Ποιο μέρος της μνήμης έχει η κάθε διεργασία
- Έχει εξειδικευμένο υλικό που παρακολουθεί τι έχει προσπελαστεί στην μνήμη και πως και επιβάλει κανόνες προστασίας

ΧΩΡΙΣ ΑΦΑΙΡΕΣΗ ΜΝΗΜΗΣ, δλδ βλέπουμε απευθείας στην φυσική μνήμη

Είναι συνήθως συστήματα που έχουν μια διεργασία στην μνήμη κάθε φορά και το ΛΣ. Η διεργασία μπορεί να είναι πολυνηματική. Επίσης μπορούμε να έχουμε και πολυπρογραμματισμό με εναλλαγή, ωστόσο θα πρέπει να φέρνουμε ολόκληρο πρόγραμμα από τον δίσκο και να ξανατοποθετήσουμε το άλλο στον δίσκο, διαδικασία χρονοβόρα και καθόλου βολική. Σε κάποιες άλλες περιπτώσεις μπορούμε να έχουμε και 2-3 διεργασίες στην μνήμη, η μνήμη χωρίζεται σε block των 2KB και έχει κλειδιά για να ξέρουμε ποιος χώρος μνήμης αντιστοιχεί που.

Η ανάγκη για επανατοποθέτηση

Όταν φέρουμε μια διεργασία στην μνήμη πρέπει να τροποποιηθούν οι εσωτερικές διευθύνσεις μνήμης ώστε να αναφέρονται στις πραγματικές διευθύνσεις. Τα προγράμματα προγραμματίζονται με το σκεπτικό ότι ξεκινάνε από τη θέση μνήμης 0, οπότε αν ένα πρόγραμμα είναι 16KB και ξεκινάει από τι θέση μνήμης 32KB θα πρέπει να επανατοποθετηθούν οι εσωτερικές διευθύνσεις.

ΧΩΡΟΙ ΔΙΕΥΘΗΝΣΕΩΝ

Ο χώρος διευθύνσεων είναι μια λογική αφαίρεση της μνήμης μιας διεργασίας. Περιέχει τις διευθύνσεις τις, δλδ τον κώδικα, τα δεδομένα(μεταβλητά και σταθερά), την στοίβα και τον σωρό.

Οι χώροι διευθύνσεων μπορούν να σχεδιαστούν ως μια μορφή δυναμικής επανατοποθέτησης. Έχουμε τους **καταχωρητές βάσεις και ορίου**. Ο καταχωρητής βάσης δίχνει την αρχική διεύθυνση όπου φορτώνεται το πρόγραμμα(διεύθυνση εκκίνησης) και ο καταχωρητής ορίου δείχνει το μέγιστο μέγεθος μνήμης που μπορεί να έχει ένα πρόγραμμα. Το πρόγραμμα φορτώνεται σε διαδοχικές θέσεις μνήμης.

Εναλλαγή διεργασιών από τη μνήμη στον δίσκο(swapping)

Αν θέλουμε να φέρουμε ένα νέο πρόγραμμα στην μνήμη και αυτό δεν χωράει τότε θα πρέπει να γίνει εναλλαγή με ένα άλλο και να πάει το άλλο στον δίσκο. Η εναλλαγή όμως μπορεί να προκαλέσει κενά στην μνήμη. Γι'αυτό και περιοδικά χρειάζεται σύμπτυξη της μνήμης, αυτό μπορεί να γίνει με αντιγραφή των διεργασιών και επανατοποθέτηση τους, αλλά δεν προτιμάται. Επίσης επειδή οι διεργασίες μπορεί να αυξάνονται δυναμικά θα πρέπει να αφήσουμε έξτρα χώρο ανάμεσα στις διεργασίες, ο οποίος όμως στην συνέχεια μπορεί να μείνει αναξιοποίητος, αλλά αν δεν είναι επαρκής θα πρέπει ολόκληρη η διεργασία να μεταφερθεί, πράγμα που έχει μεγάλο κόστος για τον επεξεργαστή.

Διαχείριση ελεύθερης μνήμης

Τεχνικές για να ξέρουμε που υπάρχει διαθέσιμη μνήμη:

- **Διαχείριση μνήμης με χάρτες bit:** διαιρούμε την μνήμη σε ισομεγέθη τμήματα και ένα bit ανά τμήμα δείχνει αν είναι διαθέσιμο(0 είναι διαθέσιμο). Μεγάλο ρόλο παίζει το πόσο μεγάλα θα είναι τα ισομεγέθη τμήματα. Αν είναι μεγάλα τότε θα υπάρχει και μεγάλη σπατάλη μνήμης και μικρός χάρτης. Αν είναι μικρά τότε θα υπάρχει μεγάλος χάρτης και μικρότερη σπατάλη μνήμης. Ωστόσο αν είναι μικρά τότε για να βρούμε ελεύθερη μνήμη για μια νέα διεργασία θα είναι χρονοβόρα διαδικασία.
- **Διαχείριση μνήμης με συνδεδεμένες λίστες:** ταξινομούμε τις οπές και διεργασίες με βάση τη διεύθυνση που ξεκινάνε. Έχουμε μια λίστα με στοιχεία για κάθε τμήμα μνήμης(τύπος(ελεύθερη ή όχι), αρχική διεύθυνση, μέγεθος). Μια διπλά συνδεδεμένη λίστα ώστε όταν ελευθερώνεται μια περιοχή μνήμης να γίνει εύκολα συγχώνευση με τα γειτονικά τμήματα μνήμης αν αυτά είναι ελεύθερα. Τώρα όσον αφορά το ποιο κομμάτι μνήμης θα επιλέξουμε για την εισερχόμενη διεργασία στην μνήμη έχουμε τις εξής επιλογές:
    1)first fit
    2)next fit: η πρώτη προσαρμογή από το σημείο που είχαμε σταματήσει
    3)best fit: μικρότερος δυνατός ελεύθερος χώρος
    4)worst fit: μεγαλύτερος δυνατός ελεύθερος χώρος
- **Ξεχωριστή λίστα για διεργασίες(μη ελεύθερη μνήμη) και χωριστή για οπές(ελεύθερη μνήμη):** Ένα πλεονέκτημα είναι ότι μπορούμε να ταξινομήσουμε τις οπές με βάση το μέγεθος τους. Υπάρχει όμως και το μειονέκτημα ότι όταν απελευθερωθεί μνήμη πρέπει να ψάξουμε σε όλη τη λίστα των οπών για να δούμε που θα εκχωρηθεί ο νέος κενός χώρος μνήμης. Γι'αυτό προτιμάται η ταξινόμηση των οπών με βάση την διεύθυνση τους. (Επίσης μπορούμε μόνο στις οπές να έχουμε δείκτες που θα συνδέουν τις κενές θέσεις μεταξύ τους.)
- **Γρήγορη προσαρμογή(quick fit):** διατηρώ πολλές λίστες οπών με βάση το μέγεθος τους και έτσι έχω εύκολη δέσμευση, αλλά υπάρχει δυσκολία στο να

συνδέσω μια περιοχή μνήμης που ελευθερώθηκε με άλλες ελεύθερες. Δεν ξέρουμε σε ποια από τις λίστες είναι οι γειτονικές οπές.

ΕΙΚΟΝΙΚΗ ΜΝΗΜΗ

Στα όσα είδαμε παραπάνω το πρόγραμμα υπήρχε όλο στην μνήμη, αλλά στα μεγάλα προγράμματα αυτό δεν είναι βολικό και η εναλλαγή είναι ακριβή. Αυτό που κάνει η εικονική μνήμη είναι να δίνει την αίσθηση στο πρόγραμμα ότι βρίσκεται όλο στην μνήμη ενώ στην πραγματικότητα ένα μεγάλο μέρος του βρίσκεται στον δίσκο και μόνο ότι χρειαζόμαστε είναι στην μνήμη. Με την εικονική μνήμη κάθε πρόγραμμα έχει δικό του χώρο διευθύνσεων ο οποίος χωρίζεται σε σελίδες σταθερού μεγέθους και κάποιες από τις σελίδες αυτές είναι στην μνήμη. Το υλικό μεταφράζει αυτόματα τις λογικές αναφορές μνήμης σε φυσικές και επίσης παγιδεύει αναφορές σε μη διαθέσιμες σελίδες(σελίδες που είναι στον δίσκο).

ΣΕΛΙΔΟΠΟΙΗΣΗ(PAGING)

Όταν χωρίζουμε το πρόγραμμα και τη μνήμη σε σταθερού μεγέθους ίσα τμήματα τότε λέμε ότι έχουμε σελιδοποίηση. Το πρόγραμμα δημιουργεί τις λογικές διευθύνσεις τις οποίες η μονάδα διαχείρισης μνήμης μέσω του πίνακα μετάφρασης τις μετατρέπει σε φυσικές. Δεν είναι απαραίτητο σελίδες του ίδιου προγράμματος στην φυσική μνήμη να είναι συνεχόμενες εφόσον έχουμε τον πίνακα μετάφρασης και ξέρουμε που είναι μια λογική σελίδα.

Πίνακας σελίδων: έχει τόσες καταχωρήσεις όσο και οι σελίδες του προγράμματος. Έχουμε και **bit παρουσίας** για να ξέρουμε αν η σελίδα είναι στην φυσική μνήμη ή στον δίσκο. Κάθε λογική διεύθυνση χωρίζεται σε δύο τμήματα:

1)αριθμός εικονικής σελίδας: πιο σημαντικά bit

2)απόσταση μέσα στη σελίδα

Το ΛΣ παρεμβαίνει μόνο όταν η σελίδα δεν είναι στην μνήμη και η μονάδα διαχείρισης μνήμης δημιουργεί μια διακοπή για να έρθει η σελίδα.

Δομή καταχώρισης πίνακα σελίδων. Έχουμε:

- **Bit προστασίας:** δλδ τι είδους πρόσβαση επιτρέπεται
- **Bit τροποποίησης:** δλδ αν η σελίδα έχει τροποποιηθεί πρόσφατα και πρέπει να γραφτεί στην μνήμη πριν απαομακρυνθεί
- **Bit αναφοράς:** αν η σελίδα έχει χρησιμοποιηθεί πρόσατα
- **Bit κρυφής μνήμς:** αν μια σελίδα δεν πρέπει να γραφτεί στην κρυφή μνήμη

ΕΠΙΤΑΧΥΝΣΗ ΣΕΛΙΔΟΠΟΙΗΣΗΣ

Όλοι οι πίνακες σελίδων αρχικά βρίσκονται όλοι στην μνήμη. Επειδή τα προγράμματα χρησιμοποιούν λίγες σελίδες σε κάθε περίοδο μπορούμε να έχουμε μια κρυφή μνήμη αναζήτησης μετάφρασης(TLB) η οποία θα έχει τις πιο συχνά χρησιμοποιούμενες σελίδες.

Τρόπος που λειτουργεί ένα σύστημα με το TLB: κάθε φορά που θέλουμε να χρησιμοποιήσουμε μια σελίδα πάμε στο TLB, αν είναι εκεί παίρνουμε την διεύθυνση μνήμης, αν όχι τότε βλέπουμε αν υπάρχει στην μνήμη και εάν υπάρχει την πάμε στην TLB αντικαθιστώντας την με κάποια άλλη και παίρνουμε την διεύθυνση της φυσικής μνήμης. Η διαδικασία αυτή γίνεται μέσω του συστήματος διαχείρισης μνήμης(MMU).

ΠΟΛΥ ΜΕΓΑΛΕΣ ΜΝΗΜΕΣ

Σε πολύ μεγάλες μνήμες μπορεί ο πίνακας σελίδων να έχει ακόμα και 1.000.000 καταχωρήσεις. Αυτό που κάνουμε σε τέτοιες περιπτώσεις είναι να φτιάχνουμε πολυεπίπεδους πίνακες διευθύνσεων. Έστω ότι οι εικονικές διευθύνσεις είναι των 32 bit. Τα 10 πρώτα bit θα αναφέρονται στην πρώτη σελίδα και θα μας πάνε σε έναν πίνακα $2^{ου}$ επιπέδου, τα επόμενα 10 bit θα μας πούνε σε ποια διεύθυνση στον πίνακα $2^{ου}$ επιπέδου να πάμε και με τα άλλα 12 bit έχουμε την τελική διεύθυνση.

Τα περισσότερα προγράμματα θέλουν λίγη μνήμη οπότε αυτό που συμβαίνει είναι ότι ελάχιστοι από τους πίνακες $2^{ου}$ επιπέδου έχουν πραγματική πληροφορία. Αυτό είναι βολικό για 32-bit συστήματα. Στα 64-bit κάνουμε κάτι άλλο. **ΑΝΕΣΤΡΑΜΜΕΝΟΙ ΠΙΝΑΚΕΣ ΣΕΛΙΔΩΝ:** ο πίνακας αυτός(aka πίνακας πλαισίων) είναι ανεξάρτητος από το πλήθος των διεργασιών και είναι ένας. Ο πίνακας εξαρτάται από το μέγεθος της φυσικής μνήμης. Ο πίνακας λέει για κάθε σελίδα ποια διεργασία την έχει δεσμεύσει. Για να βρούμε μια σελίδα χρησιμοποιείται πρώτα TLB που έχει τις συχνά χρησιμοποιούμενες σελίδες και μόνο αν δεν την βρούμε εκεί ψάχνουμε σε όλον τον ανεστραμμένο πίνακα. Για να αποφύγουμε να ψάξουμε όλον τον ανεστραμμένο πίνακα χρησιμοποιούμε έναν πίνακα κατακερματισμού στον οποίον παίρνουμε την εικονική διεύθυνση, την κατακερματίζουμε και κοιτάμε την αντίστοιχη θέση του πίνακα κατακερματισμού ο οποίος μας λέει αν υπάρχει στην μνήμη η αντίστοιχη σελίδα.

ΑΛΓΟΡΙΘΜΟΙ ΑΝΤΙΚΑΤΑΣΤΑΣΗΣ ΣΕΛΙΔΩΝ

Αντικατάσταση σελίδας από τη μνήμη κάνουμε όταν έχουμε σφάλμα σελίδας, δλδ μια σελίδα που θέλουμε να χρησιμοποιήσουμε δεν είναι στην μνήμη, αλλά στον δίσκο. Όταν απομακρύνουμε μια σελίδα πρέπει να δούμε αν έχει τροποποιηθεί ώστε αν ναι να την γράψουμε στον δίσκο. Στόχος είναι να απομακρύνουμε μια «αχρησιμοποίητη σελίδα».

- **ΒΕΛΤΙΣΟΣ ΑΛΓΟΡΙΘΜΟΣ ΑΝΤΙΚΑΤΑΣΤΑΣΗΣ:** είναι βέλτιστος μόνο στην θεωρεία. Βασίζεται στο ότι επιλέγουμε για αντικατάσταση την σελίδα που θα χρησιμοποιηθεί πιο μακριά στο μέλλον.

- **ΑΛΓΟΡΙΘΜΟΣ NRU(not recently used):** προτιμάμε να αντικαταστήσουμε σελίδες οι οποίες δεν έχουν χρησιμοποιηθεί πρόσφατα. Ο αλγόριθμος αυτός λειτουργεί με τη βοήθεια δυο bit καταστάσεων, το A(access) που δείχνει έχει γίνει πρόσφατα αναφορά στην σελίδα και το T(transform) που δείχνει αν έχουμε γράψει στην σελίδα. Μια σελίδα προς αντικατάσταση μπορεί να ανήκει σε μια από τις τέσσερεις παρακάτω κατηγορίες:
    0) Δεν έγινε αναφορά, δεν τροποποιήθηκε
    1) Δεν έγινε αναφορά, τροποποιήθηκε
    2) Έγινε αναφορά, δεν τροποποιήθηκε
    3) Έγινε αναφορά, τροποποιήθηκε

  Προτιμάμε να αφαιρέσουμε τις παλιές σελίδες, δλδ αυτές που δεν έχει γίνει πρόσφατα αναφορά(ανά κάποιο χρονικό διάστημα το ΛΣ μηδενίζει το bit A).

- **ΑΛΓΟΡΙΘΜΟΣ FIFO:** ταξινομούμε τις σελίδες με τη σειρά φόρτωσης. Αφαιρούμε αυτήν που είναι την περισσότερη ώρα στην μνήμη και βρίσκεται στην αρχή τη ουράς. Αυτό όμως έχει τον κίνδυνο να αφαιρέσουμε την σελίδα που χρησιμοποιείται πιο συχνά.

- **ΑΛΓΟΡΙΘΜΟΣ ΔΕΥΤΕΡΗΣ ΕΥΚΑΙΡΙΑΣ:** παραλλαγή FIFO. Ξεκινάμε όπως στην fifo και πάμε στην αρχή της ουράς, αν δούμε ότι η σελίδα έχει χρησιμοποιηθεί πρόσφατα(δλδ bit A είναι 1) τότε της δίνουμε μια δεύτερη ευκαιρεία μηδενίζοντας το bit A και βάζοντας την στο τέλος. Το κάνουμε αυτό μέχρι να βρούμε σελίδα με bit A 0.

- **ΑΛΓΟΡΙΘΜΟΣ ΤΟΥ ΡΟΛΟΓΙΟΥ:** παραλλαγή του αλγορίθμου δεύτερης ευκαιρίας. Αντί να έχουμε ουρά έχουμε κυκλική λίστα και κάνουμε ότι στον άλλο αλγόριθμο με τη διαφορά ότι αν έχει η σελίδα bit A 1 τότε προχωράμε τον δείκτη που δείχνει στις σελίδες στην επόμενη.

- **ΑΛΓΟΡΙΘΜΟΣ LRU(least recently used):** σκοπός είναι να αντικαταστήσουμε την σελίδα αυτήν που έχει τον περισσότερο χρόνο να προσπελαστεί. Στην υλοποίηση αυτό έχει απαγορευτικό κόστος και γι'αυτό έχουν υλοποιηθεί

προσεγγίσεις. **Πρώτη παραλλαγή:** να έχουμε έναν μετρητή ο οποίος μετά από κάθε εντολή θα αυξάνεται κατά 1. Σε κάθε αναφορά στην μνήμη ο μετρητής αυτός θα πρέπει να αποθηκεύεται για να μας δείξει πόσο πρόσφατα χρησιμοποιήθηκε η σελίδα. Η αποθήκευση θα πρέπει να γίνει στην TLB, αλλά ένας 64-Bit μετρητής θα θέλει έξτρα χώρο στην TLB κάτι που είναι δύσκολο να δοθεί. Θα μπορούσαμε να έχουμε μικρότερους μετρητές οι οποίοι ωστόσο μετά από κάποιο διάστημα θα ξεκινούσαν πάλι από το 0 λόγο του μεγέθους τους και έτσι δεν θα ξέραμε πραγματικά ποια σελίδα χρησιμοποιήθηκε το λιγότερο πρόσφατα. **Δεύτερη παραλλαγή:** για n σελίδες να έχουμε έναν n*n πίνακα όπου στην αρχή όλες οι θέσεις έχουν τιμή 0. Όταν χρησιμοποιηθεί η k σελίδα(οποιαδήποτε σελίδα) αυτό που κάνουμε είναι να βάλουμε 1 σε όλη την γραμμή k και 0 σε όλη την στήλη k. Έτσι όταν θέλουμε να κάνουμε αντικατάσταση επιλέγουμε την σελίδα που έχει στην στήλη της τα περισσότερα 1. Το μειονέκτημα σ'αυτό είναι ότι αν έχουμε πολλές σελίδες θα χρειαστούμε μεγάλο χώρο στην μνήμη, άρα είναι προτιμότερο όταν έχουμε λίγες σελίδες.

- **ΠΡΟΣΟΜΟΙΩΣΗ LRU:** έχουμε τον **αλγόριθμο NFU(not frequently used).** Αυτό που κάνουμε είναι ότι για κάθε σελίδα έχουμε έναν μετρητή. Σε κάθε διακοπή ρολογιού προσθέτουμε στον μετρητή την τιμή του bit A, που δείχνει αν η σελίδα χρησιμοποιήθηκε πρόσφατα ή όχι και στη συνέχεια μηδενίζουμε το bit A. Αυτό που πετυχαίνουμε είναι να έχουμε έναν μετρητή που σχετίζεται με τον χρόνο που έχει χρησιμοποιηθεί η διεργασία. Το μειονέκτημα με αυτό είναι ότι δεν ξεχνάει, δλδ αν μια σελίδα έχει πάρα πολύ ώρα να χρησιμοποιηθεί, αλλά στο παρελθόν χρησιμοποιήθηκε για μεγάλο διάστημα θα έχει μεγάλη τιμή στον μετρητή της και δεν θα είναι υποψήφια για αντικατάσταση σε αντίθεση με μια σελίδα που έχει μικρή τιμή μετρητή, αλλά χρησιμοποιήθηκε πολύ πρόσφατα. Οπότε χρειαζόμαστε να κρατήσουμε και μια ηλικία για το πότε τελευταία χρησιμοποιήθηκε η σελίδα. Αυτό το πετυχένουμε με την **γήρανση(aging)** των μετρητών. Σε κάθε διακοπή ρολογιού κάνουμε shift προς τα δεξιά και έπειτα προσθέτουμε το bit A από τα αριστερά.

- **ΣΥΝΟΛΟ ΕΡΓΑΣΙΑΣ:** όταν ξεκινάει μια διεργασία δεν έχει καμία σελίδα στην μνήμη και έτσι στην αρχή έχουμε πολλά σφάλματα σελίδων. Οι σελίδες που χρησιμοποιεί μια διεργασία μια συγκεκριμένη χρονική περίοδο λέγεται **σύνολο εργασίας(working set)**. Σκοπός είναι κάθε φορά να βρίσκεται στην μνήμη το σύνολο εργασίας. Θέλουμε να φέρνουμε σελίδες που ανήκουν στο σύνολο εργασίας ώστε να αποφεύγουμε σφάλματα σελίδων, τα οποία σφάλματα αν είναι πολύ συχνά αποτρέπουν την διεργασία να τρέξει κανονικά αφού αφιερώνεται πολύς χρόνος στις αντικαταστάσεις σελίδων.

Οι διεργασίες μετά από κάποιο διάστημα διακόπτονται(χρονοπρογραμματισμός) για να εκτελεστούν και άλλες διεργασίες. Όταν επανέλθει μια διεργασία στην μνήμη μπορούμε να μην κάνουμε τίποτα και να έχουμε σφάλματα σελίδων στην αρχή. Μπορούμε όμως να παρακολουθήσουμε το σύνολο εργασίας και να φορτώσουμε αυτό όταν ξαναξεκινήσει η διεργασία μετά την διακοπή(prepaging/προσελιδοποίηση). Μπορούμε να παρακολουθήσουμε το σύνολο εργασιών βλέποντας το πλήθος αναφορών που έχουν γίνει για παράδειγμα στα τελευταία 100ms.

- **ΑΛΓΟΡΙΘΜΟΣ ΣΥΝΟΛΟΥ ΕΡΓΑΣΙΑΣ:** έχουμε δυο πεδία, το bit A και την τελευταία προσπέλαση, σε κάθε κύκλο ρολογιού αυξάνουμε τον χρόνο κατά 1, οπότε σε μια διακοπή αν το bit A είναι 1σημειόνουμε τον τρέχοντα εικονικό χρόνο και όταν αντικαταστήσουμε μια σελίδα θα φύγει αυτή που έχει τον περισσότερο χρόνο να προσπελαστεί. Υπολογίζουμε την ηλικία-> τρέχον χρόνος-χρόνος τελευταίας προσπέλασης. Αν η ηλικία είναι πάνω από ένα συγκεκριμένο όριο τότε αντικαθιστούμε όλες αυτές τις σελίδες.

- **ΑΛΓΟΡΙΘΜΟΣ WSCLOCK:** είναι μια ένωση του αλγορίθμου συνόλου εργασίας και του αλγορίθμου ρολογιού. Οι σελίδες οργανώνονται σε κυκλική λίστα, σε κάθε σφάλμα ξεκινάμε από κει που είχαμε μείνει. Αν η σελίδα έχει A=1 προχωράμε και την μηδενίζουμε, αν είναι 0 και είναι η ηλικία της μεγαλύτερη από το όριο που είχαμε δώσει την αντικαθιστούμε, αν είναι 0 και είναι καθαρή(το T είναι 0) την αντικαθιστούμε, αν δεν είναι καθαρή την προγραμματίζουμε για εγγραφή και προχωράμε στην επόμενη. Όταν στείλουμε n σελίδες στον δίσκο να σταματάμε αλλά δεν θέλουμε να αδειάσουμε την μνήμη.

Οι αλγόριθμοι γήρανσης(προσέγγιση LRU) και wsclock(προσέγγιση συνόλου εργασίας) είναι πιο πρακτικοί.

ΘΕΜΑΤΑ ΣΧΕΔΙΑΣΜΟΥ

ΤΟΠΙΚΕΣ ΚΑΙ ΚΑΘΟΛΙΚΕΣ ΠΟΛΙΤΙΚΕΣ

Οι **τοπικές πολιτικές** λένε ότι η αντικατάσταση σελίδων γίνεται μόνο με τις άλλες σελίδες τις διεργασίας και το μέρος της μνήμης που καταλαμβάνει μια διεργασία από την αρχή δεν αλλάζει.

Οι **καθολικές πολιτικές** λένε ότι η αντικατάσταση σελίδων μπορεί να γίνει και με σελίδες από άλλες διεργασίες και το μέρος της μνήμης που έχει στη διάθεση της μια διεργασία μπορεί να αλλάξει δυναμικά. Οι καθολικές πολιτικές λειτουργούν καλύτερα γιατί

μπορούμε να μεταβάλουμε το μέρος μνήμης που καταλαμβάνει μια διεργασία. Η αρχική κατανομή της μνήμης στις διεργασίας μπορεί να γίνει είτε δίκαια δλδ αν έχουμε n διεργασίας να τους δώσουμε 1/n χώρο της μνήμης είτε να τους δώσουμε χώρο ανάλογα με το σύνολο εργασίας που χρειάζεται. Αυτό που πρέπει να προσέξουμε είναι όταν αντικαθιστούμε σελίδες να μην αφαιρούμε συνέχεια της ίδιας διεργασίας για να μην φτάσει αυτή σε σημείο να μην μπορεί να εκτελεστεί. Θα πρέπει κάθε διεργασία να έχει τουλάχιστον ένα ελάχιστο πλήθος σελίδων στην μνήμη.

Για να δούμε αν μια διεργασία χρειάζεται περισσότερη μνήμη χρησιμοποιούμε τον **αλγόριθμο συχνότητας σφαλμάτων σελίδων(PFF)**, ο οποίος παρακολουθεί τα σφάλματα σελίδων που κάνει μια διεργασία, αν αυτά είναι πολλά σημαίνει ότι χρειάζεται περισσότερες σελίδες στην μνήμη, αν αυτά είναι λίγα σημαίνει ότι έχει παραπάνω σελίδες, μνήμη από ότι χρειάζεται.

ΕΛΕΓΧΟΣ ΦΟΡΤΙΟΥ

Αναφέρεται στον αν δεν υπάρχει αρκετός χώρος στην μνήμη ώστε οι διεργασίες να έχουν το σύνολο εργασίας τους στην μνήμη, δλδ το PFF όλων τον διεργασιών να είναι υψηλό. Η λύση είναι να πάμε μερικές διεργασίες στον δίσκο(εξάλειψη από την μνήμη). Πρέπει να μεταφέρουμε μερικές διεργασίες και όχι σχεδόν όλες για να αξιοποιούνται όλοι οι πόροι του συστήματος.

ΜΕΓΕΘΟΣ ΣΕΛΙΔΑΣ

Το υλικό μας παρέχει ένα βασικό μέγεθος σελίδας και το λογισμικό μπορεί να λειτουργήσει με πολλαπλάσια της. Έχουμε στην διάθεση μας δύο επιλογές για το μέγεθος σελίδας, μικρές ή μεγάλες. Οι **μεγάλες σελίδες** έχουν το αρνητικό ότι οι σελίδες μπορεί να μην είναι πολύ γεμάτες και έτσι ένας χώρος μνήμης μένει αναξιοποίητος, αλλά έχουν το θετικό ότι μας συμφέρει να μεταφέρουμε λίγες μεγάλες σελίδες(το κόστος μεταφοράς μεγάλων ή μικρών σελίδων είναι σχεδόν ίδιος). Οι **μικρές σελίδες** έχουν το αρνητικό ότι απαιτούν μεγάλο πίνακα σελίδων, αλλά το θετικό είναι στη σελίδα ελάχιστος χώρος μνήμης μπορεί να μείνει αχρησιμοποίητος.

ΕΝΤΟΛΕΣ ΚΑΙ ΔΕΔΟΜΕΝΑ

Μια άλλη επιλογή που έχουμε να κάνουμε είναι αν τα δεδομένα και ο κώδικας θα βρίσκονται στον ίδιο χώρο μνήμης. Συνήθως βρίσκονται σε ενιαίο χώρο, αλλά αν θέλαμε θα μπορούσαμε να έχουμε έναν χωριστό χώρο διευθύνσεων για τις εντολές και έναν άλλον χωριστό για τα δεδομένα.

ΚΟΙΝΟΧΡΗΣΤΕΣ ΣΕΛΙΔΕΣ

Ένα πρόγραμμα μπορεί να τρέχει από πολλές διεργασίες οπότε το κομμάτι εντολών είναι ίδιο για όλες, αυτό που διαφέρει είναι το κομμάτι δεδομένων. Οπότε θα μας σύμφερε το κοινό αυτό κομμάτι να το είχαμε μια φορά για όλες τις διεργασίες που το χρησιμοποιούν. Έχουμε για κάθε διεργασία έναν πίνακα σελίδων για τα δεδομένα και έναν άλλον πίνακα σελίδων για τις εντολές ο οποίος όμως είναι κοινόχρηστος από πολλές διεργασίες.

ΚΟΙΝΟΧΡΗΣΤΕΣ ΒΙΒΛΙΟΘΗΚΕΣ

Έχουμε τα dynamic linked libraries(DLL) που η σύνδεση μαζί τους από την διεργασία γίνεται μόνο τη στιγμή που θα χρειαστεί η χρήση τους. Φορτώνουμε όλο το DLL στην μνήμη και όχι μόνο τις συναρτήσεις που χρησιμοποιούνται και έτσι όλες οι διεργασίες έχουν πρόσβαση στη βιβλιοθήκη.

ΧΑΡΤΟΓΡΑΦΗΜΕΝΑ ΑΡΧΕΙΑ

Δίνεται η αίσθηση ότι είναι στην μνήμη αλλά φορτώνεται όπως οι διεργασίες με σελιδοποίηση.

ΠΟΛΙΤΙΚΗ ΚΑΘΑΡΙΣΜΟΥ

Μια σελίδα την οποία πάμε να την αφαιρέσουμε από την μνήμη και έχει τροποποιηθεί το περιεχόμενο της θα πρέπει να την γράψουμε πρώτα στον δίσκο και αυτό θέλει κάποιο χρόνο. Για να μην περιμένουμε να γραφτεί μια σελίδα στον δίσκο και μετά να φέρουμε την καινούρια έχουμε τον **δαίμονα αντικατάστασης** ο οποίος φροντίσει να υπάρχει πάντα μια καθαρή σελίδα προς αντικατάσταση. Αυτό που κάνει είναι να ξυπνάει περιοδικά και να ελέγχει σελίδες που είναι υποψήφιες προς αντικατάσταση και από αυτές όσες είναι τροποποιημένες να τις βάζει σε μια ουρά για εγγραφή στον δίσκο, όμως δεν διώχνει τις σελίδες απλά τις καθαρίζει.

ΖΗΤΗΜΑΤΑ ΥΛΟΠΟΙΗΣΗΣ

ΡΟΛΟΣ ΤΟΥ ΛΣ

Όταν ξεκινήσει μια διεργασία το ΛΣ πρέπει να εκτιμήσει το μέγεθος του προγράμματος και των δεδομένων. Ασχολείται με τον χρονοπρογραμματισμό της διεργασίας, με τα σφάλματα σελίδας και με τον τερματισμό της διεργασίας.

ΧΕΙΡΙΣΜΟΣ ΣΦΑΛΜΑΤΩΝ ΣΕΛΙΔΑΣ

Όταν έχουμε σφάλμα σελίδας το υλικό προκαλεί παγίδα στον πυρήνα και στην συνέχεια αποθηκεύεται ο μετρητής προγράμματος και άλλοι καταχωρητές ώστε να ξέρουμε την κατάσταση που βρίσκεται η διεργασία. Μέσω του χειριστή σφαλμάτων σελίδας πρέπει να βρούμε ποια σελίδα μας ζητήθηκε να φέρουμε. Αφού βρούμε την σελίδα που θα φέρουμε στην μνήμη θα κάνουμε έλεγχο εγκυρότητας και πρόσβασης για την σελίδα αυτή, δλδ αν όντως ανήκει στον εικονικό χώρο της διεργασίας. Αν όλα καλά πρέπει να βρούμε μια σελίδα που είναι στην μνήμη για αντικατάσταση. Αν η σελίδα για αντικατάσταση είναι «βρώμικη» τότε αναστέλλεται η διεργασία για να γραφτεί η σελίδα στον δίσκο και συνεχίζουμε μια άλλη διεργασία. Αφού ολοκληρωθεί η εγγραφή της σελίδας στον δίσκο θα γίνει η μεταφορά στην μνήμη της σελίδας που θέλουμε και θα ενημερωθεί ο πίνακας σελίδων και το πρόγραμμα θα ξεκινήσει από εκεί που δίνει ο μετρητής του προγράμματος και θα επαναφέρουμε στην μνήμη τους άλλου καταχωρητές στην κατάσταση που ήταν πριν την διακοπή.

ΑΝΤΙΓΡΑΦΑ ΑΣΦΑΛΕΙΑΣ ΕΝΤΟΛΩΝ

Η επανεκτέλεση της εντολής η οποία προκάλεσε σφάλμα σελίδας και την διακοπή δεν είναι τόσο απλή. Για να καταλάβουμε πότε δημιουργήθηκε η διακοπή έχουμε κάποιους ειδικούς καταχωρητές που κρατάνε την αρχική διεύθυνση της τρέχουσας εντολής και τις αλλαγές που έγιναν σε καταχωρητές και χρησιμοποιούνται για να βρούμε την αιτία του σφάλματος και να ξεκινήσουμε την εντολή από την αρχή κάνοντας αναιρέσεις των αλλαγών.

ΚΛΕΙΔΩΜΑ ΣΕΛΙΔΩΝ ΣΤΗΝ ΜΝΗΜΗ

Αν μια διεργασία ξεκινά ανάγνωση του δίσκου, τότε η διεργασία μπλοκάρεται περιμένοντας τα δεδομένα και επειδή υπάρχει κίνδυνος την σελίδα αυτή να την πάρει μια άλλη διεργασία την μπλοκάρουμε.

ΔΕΥΤΕΡΕΥΟΥΣΑ ΜΝΗΜΗ

Μπορεί να είναι ο δίσκος. Μπορούμε να έχουμε έναν ειδικό δίσκο εναλλαγής αποκλειστικά και μόνο για αντικατάσταση σελίδων. Για κάθε διεργασία αρκεί να ξέρουμε που ξεκινάει στον δίσκο και ποιο το μέγεθος της.

ΤΜΗΑΜΤΟΠΟΙΗΣΗ(SEGMENATATION)

Ας πάρουμε για παράδειγμα τους μεταγλωττιστές που αποτελούνται από πολλά τμήματα(κώδικας, δεδομένα….) τα οποία επειδή μπορεί να επεκταθούν κατά την εκτέλεση πρέπει να έχουν κενούς χώρους μνήμης μεταξύ τους. Επειδή δεν ξέρουμε εξαρχής πόσο χώρο θα καταναλώσει κάθε τμήμα μπορεί κατά την εκτέλεση να προκύψουν θέματα. Αυτό που μπορούμε να κάνουμε είναι να θεωρήσουμε το κάθε τμήμα τις διεργασίας ως ένα ξεχωριστό χώρο διευθύνσεων και να έχουμε πολλούς εικονικούς χώρους διευθύνσεων. Αυτό το κάνει η **τμηματοποίηση.** Στην τμηματοποίηση κάθε διεργασία αποτελείται από τμήματα και κάθε τμήμα είναι μια γραμμική ακολουθία διευθύνσεων. Τα τμήματα έχουν διαφορετικό και μεταβλητό μέγεθος και υπάρχει λογικός καταμερισμός της μνήμης.

Πλεονεκτήματα τμηματοποίησης:

- Απλούστερη σύνδεση διαδικασιών αφού βρίσκονται σε διαφορετικά τμήματα.
- Διευκόλυνση κοινής χρήσης βιβλιοθηκών. Κάθε βιβλιοθήκη είναι και ένα διαφορετικό τμήμα και οι διεργασίες την βλέπουν με τον ίδιο τρόπο.
- Προστατεύουμε τις διεργασίες σε επίπεδο τμημάτων, οπότε διαφορετικά τμήματα μπορούν να έχουν διαφορετική προστασία.

PAGING VS SEGMENTATION

Χρειάζεται να είναι ενήμερος ο προγραμματιστής ότι χρησιμοποιείται η συγκεκριμένη τεχνική;	Όχι	Ναι
Πόσοι γραμμικοί χώροι διευθύνσεων υπάρχουν;	1	Πολλοί
Μπορεί ο συνολικός χώρος διευθύνσεων να υπερβαίνει το μέγεθος της φυσικής μνήμης;	Ναι	Ναι
Μπορούν οι διαδικασίες και τα δεδομένα να είναι διακριτά και να προστατεύονται ξεχωριστά;	Όχι	Ναι
Είνα εύκολος ο χειρισμός πινάκων μεταβλητού μεγέθους;	Όχι	Ναι
Διευκολύνεται η κοινή χρήση των διαδικασιών ανάμεσα στους χρήστες;	Όχι	Ναι
Γιατί επινοήθηκε αυτή η τεχνική;	Για να δημιουργηθεί μεγάλος και γραμμικός χώρος διευθύνσεων χωρίς να χρειάζεται να επεκταθεί η φυσική μνήμη	Για να επιτραπεί στα προγράμματα και τα δεδομένα να χωρίζονται σε λογικά ανεξάρτητους χώρους διευθύνσεων, και να υποβοηθείται η κοινοχρησία και η προστασία

ΥΛΟΠΟΙΗΣΗ ΤΜΗΜΑΤΟΠΟΙΗΣΗΣ

Τα τμήματα έχουν μεταβλητό μέγεθος και η φόρτωση αφαίρεση τμημάτων μπορεί να αφήσει κενά στην μνήμη.

ΥΛΟΠΟΙΗΣΗ: MULTICS

Το σκεπτικό εδώ είναι ότι κάθε τμήμα μπορεί να σελιδοποιηθεί και έτσι έχουμε φόρτωση μέρους των μεγάλων τμημάτων στην μνήμη και έχουμε απλούστερη διαχείριση της μνήμης με σελίδες.

Κάθε διεργασία έχει έναν πίνακα τμημάτων. Ο πίνακας αυτός για κάθε τμήμα έχει έναν περιγραφέα(discriptor). Ο πίνακας όντας και αυτός τμήμα σελιδοποιείται. Ο περιγραφέας του τμήματος δείχνει στον πίνακα σελίδων του τμήματος.

Στην διεργασία μπορούν να συμβούν διάφορα σφάλματα όπως προστασίας, τμήματος ή σελίδας.

Για να επιταχύνουμε την διαδικασία εύρεσης μιας σελίδας χρησιμοποιούμε τους πίνακες TLB που βρίσκονται στην κρυφή μνήμη.

ΥΛΟΠΟΙΗΣΗ: PENTIUM

Στον Intel Pentium γίνεται η σύγχρονη εφαρμογή της τμηματοποίησης με σελιδοποίηση.

Κάθε διεργασία βλέπει σε δύο πίνακες τμημάτων:

1)στον τοπικό πίνακα τμημάτων ο οποίος είναι ένας ανά διεργασία και περιέχει κώδικα και δεδομένα

2)στον καθολικό πίνακα διεργασιών που είναι κοινός για όλες και περιέχει το ΛΣ και την μνήμη.

Ο Pentium περιέχει 6 καταχωρητές τμημάτων οι οποίοι περιέχουν έναν **επιλογές(descriptor)** που δείχνει σε ένα τμήμα. Μαζί με τον επιλογές φορτώνεται και ο περιγραφέας. Και χρησιμοποιούμε TLB για επιτάχυνση σελιδοποίησης.

Αν θέλουμε στον Pentium μπορούμε να αγνοήσουμε την τμηματοποίηση και να έχουμε μόνο σελιδοποίηση, στην ουσία όλα τα τμήματα που αποτελούν μια διεργασία βρίσκονται πλέον σε ένα τμήμα.

ΕΝΟΤΗΤΑ 4: ΣΥΣΤΗΜΑΤΑ ΑΡΧΕΙΩΝ

Γιατί χρειαζόμαστε τα μέσα αποθήκευσης;

- Για αποθήκευση μεγάλου όγκου πληροφοριών, ακόμα και μεγαλύτερου από της μνήμης της διεργασίας.
- Για ανεξαρτητοποίηση πληροφοριών από διεργασίες. Οι πληροφορίες παραμένουν μόνιμα στα αποθηκευτικά μέσα ακόμα και όταν αποτύχει ή τερματίσει η διεργασία ή ακόμα και όταν κλείσουμε το PC.
- Για ταυτόχρονη προσπέλαση πληροφοριών. Η εικονική μνήμη είναι ιδιωτική σε κάθε διεργασία.

Τα μέσα αποθήκευσης όπως τα θεωρεί το ΛΣ είναι μια γραμμική αλληλουχία μπλοκ σταθερού μεγέθους και πάνω τους κάνουμε δύο βασικές λειτουργίες, εγγραφή και ανάγνωση μπλοκ.

Το **σύστημα αρχείων** εφαρμόζει μια λογική δομή πάνω από ένα μέσω αποθήκευσης ώστε η διεργασία αντί να βλέπει μια σειρά από αριθμημένα block βλέπει αρχεία με ονόματα οργανωμένα σε καταλόγους όπου το κάθε αρχείο αποτελείται από μια ακολουθία πληροφοριών και έχει δική του προστασία.

ΑΡΧΕΙΑ

ΟΝΟΜΑΣΙΑ ΑΡΧΕΙΩΝ

Κάθε αρχείο για το ΛΣ είναι ένα αντικείμενο(μια οντότητα). Οι διεργασίες δεν αναφέρονται σε block μνήμης αλλά σε ονόματα αρχείων. Η αναφορά σε ονόματα αρχείων στο UNIX είναι case sensitive ενώ στο MS-DOS είναι case insensitive. Κάθε αρχείο έχει ένα όνομα, μια τελεία και μετά μια προέκταση η οποία προσδιορίζει το είδος του αρχείου. Στο UNIX δεν είναι απαραίτητο να έχουμε προέκταση ή μπορούμε να έχουμε ακόμα και δύο, ενώ στα WINDOWS έχει σημασία, αν π.χ έχουμε κατάληξη .exe(που είναι δεσμευμένη από το ΛΣ και δεν μπορούμε να αλλάξουμε τον τρόπο που θα εκτελεστεί) σημαίνει ότι είναι εκτελέσιμο και πρέπει να ανοίξει με συγκεκριμένο τρόπο. Άλλα αρχεία βέβαια όπως τα .pdf που είναι του Acrobat Reader μπορεί ο προγραμματιστής ή ο χρήστης να τα ορίσει να ανοίξουν με ένα άλλο συμβατό πρόγραμμα.

ΔΟΜΗ ΑΡΧΕΙΩΝ

Έχουμε τρείς τρόπους με τους οποίους μπορούν να δομούνται τα αρχεία:

- **Ακολουθία Bytes χωρίς εσωτερική δομή.** Είναι το πιο απλό αλλά και πιο ευρέως χρησιμοποιούμενο αφού επιτρέπει στις διεργασίες να χειριστούν τα αρχεία όπως θέλουν, είναι πιο ευέλικτα.
- **Ακολουθία εγγραφών σταθερού μήκους.**

- **Δέντρα εγγραφών με κλειδιά.** Έχουμε πολλές εγγραφές και τις αναζητούμε με βάση κάποιο κλειδί. Είναι εύκολη η εισαγωγή νέας εγγραφής. Έχει ωστόσο μεγάλο κόστος και είναι πιο περίπλοκο στην υλοποίηση, γι'αυτό ακόμα και βάσεις δεδομένων που κάνουν αναζητήσει με βάση κάποιο κλειδί χρησιμοποιούν την ακολουθία bytes χωρίς εσωτερική δομή και έχουν και άλλα αρχεία για να πετυχαίνουν την γρήγορη αναζήτηση και άλλες λειτουργίες.

ΤΥΠΟΙ ΑΡΧΕΙΩΝ

- **Κανονικά αρχεία:** περιέχουν πληροφορίες χρηστών, όπως τα **αρχεία κειμένου ASCII** και τα **δυαδικά αρχεία.**
- **Κατάλογοι:** αρχεία ειδικής μορφής για οργάνωση.
- **Ειδικά αρχεία χαρακτήρων:** σειριακές συσκευές.
- **Ειδικά αρχεία μπλοκ:** συσκευές αποθήκευσης.

Ένα βασικό ερώτημα είναι αν θα πρέπει το ΛΣ να γνωρίζει τους τύπους αρχείων. Σίγουρα πρέπει να αναγνωριστούν τα εκτελέσιμα και ό,τι αρχείο σχετίζεται με το ΛΣ. Για άλλα αρχεία όπως π.χ .jpeg είναι καλύτερο να αφήνεται στον προγραμματιστή η επιλογή του πως θα ανοίξει τα αρχεία.

ΠΡΟΣΒΑΣΗ ΣΤΑ ΑΡΧΕΙΑ

- **Σειριακή προσπέλαση:** μπορεί να γίνει ανά byte(ή ανά πολλά bytes) ή ανά εγγραφή και μπορούμε οποιαδήποτε στιγμή να πάμε στην αρχή του αρχείου.
- **Τυχαία προσπέλαση:** είναι απαραίτητη σε πολλές εφαρμογές και ειδικά σε βάσεις δεομένων. Εφικτό σε δίσκους, αλλά όχι σε ταινίες(διαβάζουν μόνο μπρος-πίσω). Μπορούμε στους δίσκους να κινήσουμε τον δείκτη σε όποιο σημείο θέλουμε.

ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΑΡΧΕΙΩΝ

Το σύστημα αρχείων για κάθε αρχείο διατηρεί κάποιες πληροφορίες(χαρακτηριστικά/μεταδεδομένα αρχείων).

- Τυποποιημένες πληροφορίες
- Πληροφορίες προστασίας
- Μέγεθος
- Χρονικά πεδία
- Σημαίες κατάστασης
- Στοιχεία εγγραφών

ΛΕΙΤΟΥΡΓΙΕΣ ΑΡΧΕΙΩΝ

- **Create**
- **Delete**
- **Open:** φέρνουμε τα χαρακτηριστικά του αρχείου στην μνήμη και πρέπει μέσω του ονόματος του να πάμε σε μια φυσική διεύθυνση στον δίσκο.
- **Close:** κλείσιμο αρχείου και εγγραφή εκκρεμών μπλοκ.
- **Read**
- **Write**
- **Append**
- **Seek:** αλλαγή θέσης ανάγνωσης/εγγραφής.
- **Get attributes**
- **Set attributes**
- **Rename**

ΚΑΤΑΛΟΓΟΙ

Οι κατάλογοι(directories) ή φάκελοι(folders) είναι μια ειδική κατηγορία αρχείων. Τα αρχεία οργανώνονται σε καταλόγους.

ΚΑΤΑΛΟΓΟΙ ΕΝΟΣ ΕΠΙΠΕΔΟΥ

Όλα τα αρχεία βρίσκονται σε ένα μόνο κατάλογο και αυτό είναι μια λογική χρήση αν υπάρχει ένας μόνο χρήστης. Όσο μεγαλώνει το σύστημα, ακόμα και σε συστήματα ενός χρήστη θα θέλαμε να έχουμε παραπάνω καταλόγους για να οργανώνουμε τα αρχεία με βάση κάποια χαρακτηριστικά. Οι κατάλογοι ενός επιπέδου χρησιμοποιούνται σε ενσωματωμένες συσκευές όπου κάνουμε συγκεκριμένες λειτουργίες και έχουμε μικρό πλήθος αρχείων.

ΙΕΡΑΡΧΙΚΟΙ ΚΑΤΑΛΟΓΟΙ

Είναι μια δεντρική δομή. Υπάρχει ο βασικός κατάλογος του συστήματος και κάθε χρήστης έχει δικό του βασικό κατάλογο και μέσα σε αυτόν μπορεί να υπάρχουν άλλοι κατάλογοι ή και αρχεία.

ΟΝΟΜΑΤΑ ΔΙΑΔΡΟΜΩΝ

Απόλυτο όνομα διαδρομής(absolute path name): ξεκινάμε από τον βασικό κατάλογο και καταλήγουμε στο αρχείο που θέλουμε. Η διαφορά στο όνομα διαδρομής σε UNIX και WINDOWS είναι ότι το unix χρησιμοποιεί slash(/) για διαχωρισμό καταλόγων, ενώ τα windows χρησιμοποιούν backslash(\). Αν έχουμε slash ή backslash στην αρχή της διεύθυνσης σημαίνει ότι αναφερόμαστε σε absolute path.

Σχετικό όνομα διαδρομής(relative path): το path αυτό προστίθεται στο όνομα του τρέχοντος καταλόγου εργασίας και βγαίνει τελικά ένα absolute path.

Τα απόλυτα ονόματα λειτουργούν παντού, ανεξάρτητα από κατάλογο εργασίας και γι'αυτό χρησιμοποιούνται και από τις βιβλιοθήκες. Τώρα να θέλουμε να κάνουμε κάτι σε τοπικό επίπεδο χρησιμοποιούμε σχετικό όνομα και φροντίσουμε να αλλάξουμε τον τρέχοντα κατάλογο σε αυτόν στον οποίο θα δουλέψουμε.

Στους καταλόγους έχουμε δύο **καταχωρήσεις . και ..**

- **.** : για να αναφερθούμε στον τρέχοντα κατάλογο
- **..** : για να αναφερθούμε στον γονικό(προηγούμενο) κατάλογο

ΛΕΙΤΟΥΡΓΙΕΣ ΚΑΤΑΛΟΓΩΝ

- **Create:** δημιουργία κενού καταλόγου.
- **Delete:** για να διαγράψουμε έναν κατάλογο πρέπει να είναι κενός, αν έχει αρχεία τότε πρέπει να διαγράψουμε πρώτα αυτά.
- **Opendir:** άνοιγμα καταλόγου για ανάγνωση.
- **Closedir:** κλείσιμο καταλόγου.
- **Readdir:** διάβασμα της επόμενης καταχώρησης.
- **Rename:** αλλαγή ονόματος καταλόγου.
- **Link:** σύνδεση ενός αρχείου με έναν κατάλογο, το ίδιο αρχείο μπορεί να εμφανιστεί σε πολλούς καταλόγους και με διαφορετικά ονόματα.
- **Unlink:** αφαίρεση αρχείου από κατάλογο, αν το αρχείο υπάρχει σε ένα μόνο κατάλογο τότε διαγράφεται.
- **Πραγματικοί σύνδεσμοι:** πολλοί κατάλογοι να δείχνουν στο ίδιο αρχείο(δομή).
- **Συμβολικοί σύνδεσμοι:** δείχνουν σε ένα ειδικό αρχείο το οποίο δείχνει με τη σειρά του σε ένα αρχείο. Λιγότερο αποδοτικό από τους πραγματικούς συνδέσμους, αλλά πιο ευέλικτο.

ΥΛΟΠΟΙΗΣΗ ΣΥΣΤΗΜΑΤΟΣ ΑΡΧΕΙΩΝ

ΔΙΑΤΑΞΗ ΣΥΣΤΗΜΑΤΟΣ ΑΡΧΕΙΩΝ

Ο δίσκος έχει στην αρχή(στον τομέα 0) μια βασική εγγραφή εκκίνησης(**MBR**). Ο δίσκος χωρίζεται σε διαμερίσματα(**partitions**) τα οποία χωρίζονται σε block. Αμέσως μετά το MBR βρίσκεται ο **πίνακας διαμερισμάτων** που λέει που ξεκινάει το κάθε διαμέρισμα και από πόσα blocks αποτελείται.

Παράδειγμα: σύστημα αρχείων τύπου UNIX

Αρχικά έχουμε ένα μπλοκ εκκίνησης(πιθανό να είναι κενό). Έχουμε το υπερμπλοκ που περιέχει πληροφορίες για το κάθε **partition**. Έχουμε πληροφορίες για τα ελεύθερα μπλοκ, Και έχουμε τους **κόμβους ι** που είναι δομές δεδομένων, μια για κάθε αρχείο ή κατάλογο και δείχνουν δείκτες προς πραγματικά δεδομένα. Τέλος έχουμε τον βασικό κατάλογο αρχείων συστήματος και τους υπόλοιπους καταλόγους και αρχεία.

ΥΛΟΠΟΙΗΣΗ ΤΩΝ ΑΡΧΕΙΩΝ

Πώς οργανώνονται τα αρχεία στον δίσκο:

- Η απλή περίπτωση είναι τα αρχεία να **κατανέμονται σε συνεχόμενα blocks:** Για να παρακολουθήσουμε ένα αρχείο αυτό που χρειαζόμαστε είναι το αρχικό block και το μήκος, είτε σε bytes, είτε σε blocks. Υπάρχει το μειονέκτημα ότι όταν διαγράψουμε ένα αρχείο επειδή ήταν όλες οι πληροφορίες του, τα δεδομένα του σε συνεχόμενα μπλοκ δημιουργούνται κενά στον δίσκο. Μπορούμε να διαχειριστούμε τις κενές περιοχές με μια **λίστα κενών περιοχών**. Τα κενά που δημιουργούνται οδηγούν σε κατάτμηση της ελεύθερου χώρου και μπορεί να μην υπάρχει αρχείο τόσο μικρό ώστε να χωρέσει σε ένα από τα τμήματα του δίσκου που είναι κενά και έτσι μπορεί αυτό να μείνει αχρησιμοποίητο για πάντα. Αν θέλαμε η τεχνική αυτή να είναι αποτελεσματική θα έπρεπε να ξέρουμε εξαρχής το μέγεθος των αρχείων, πράγμα αδύνατο. Η τεχνική αυτή χρησιμοποιείται σε συστήματα όπου οι συσκευές έχουν ένα σταθερό μέγεθος και δεν αλλάζουν, π.χ DVD's.
- **Κατανομή block με συνδεδεμένη λίστα**: ο κατάλογος στον οποίο περιέχεται το αρχείο μας δείχνει το πρώτο μπλοκ του αρχείου. Στην αρχή κάθε μπλοκ έχουμε δείκτη προς το επόμενο block. Η τυχαία προσπέλαση γίνεται διασχίζοντας τη λίστα. Έτσι δεν έχουμε προβλήματα κατάτμησης ελεύθερου χώρου. Αντί να έχουμε τους δείκτες στην αρχή κάθε μπλοκ μπορούμε να έχουμε έναν **πίνακα κατανομής αρχείων(File Allocation Table(FAT))** στον οποίο βρίσκεται η συνδεδεμένη λίστα. Το μέγεθος του πίνακα είναι όσο το πλήθος των blocks. Υπάρχει και μια άλλη εναλλακτική λύση, οι **κόμβοι i(index node).** Σε κάθε αρχείο αντιστοιχεί και ένας κόμβος i. Αν δεν επαρκεί ο χώρος στον κόμβο ι για να έχουμε όλους τους

δείκτες του αρχείου τότε έχουμε μια πρόσθετη δομή(πρόσθετα μπλοκ δεικτών) που περιέχει τους υπόλοιπους κόμβους.

ΥΛΟΠΟΙΗΣΗ ΤΩΝ ΚΑΤΑΛΟΓΩΝ

Ο τρόπος που αποθηκεύονται οι κατάλογοι στον δίσκο είναι ίδιος με αυτόν που αποθηκεύονται τα αρχεία, αφού οι κατάλογοι είναι ειδική κατηγορία αρχείων.

Η εσωτερική τους δομή μπορεί να είναι η εξής:

- Έχουμε έναν πίνακα που έχει καταχωρήσεις σταθερού μεγέθους και αποτελείται από το όνομα του αρχείου και τα χαρακτηριστικά του και χρησιμοποιείται σε συστήματα τύπου FAT.
- Έχουμε έναν πίνακα που έχει καταχωρήσεις σταθερού μεγέθους που έχει το όνομα του καταλόγου και έναν δείκτη προς μια δομή που περιέχει τα χαρακτηριστικά του καταλόγου(index nodes).

Αναζήτηση ονομάτων σε καταλόγους:

- Γραμμική αναζήτηση: πρέπει μα διαβάζουμε μέχρι το τέλος για να βρούμε το όνομα και αυτό απαιτεί αρκετό χρόνο ειδικά σε μεγάλους καταλόγους.
- Για τους μεγάλους καταλόγους είναι προτιμότερος ο κατακερματισμός. Χρησιμοποιούμε έναν σταθερού μεγέθους πίνακα κατακερματισμού που βρίσκεται στην αρχή του καταλόγου.
- Κρυφή μνήμη για αποφυγή αναζητήσεων. Βρίσκονται οι πιο συχνά χρησιμοποιούμενοι κατάλογοι κα αρχεία.

ΚΟΙΝΟΧΡΗΣΤΑ ΑΡΧΕΙΑ

Εμφανίζονται σε πολλούς καταλόγους χρηστών. Πολλοί δείκτες προς ένα αρχείο(DAG). Υλοποίηση κοινόχρηστων αρχείων:

- **Απλοί σύνδεσμοι:** οι δείκτες βρίσκονται σε index node. Κάθε κατάλογος έχει σύνδεσμο προς το αρχείο και περιέχει τις πληροφορίες γι'αυτό. Αν ωστόσο τροποποιηθεί σε έναν κατάλογο το αρχείο θα πρέπει να ενημερωθούν όλοι για αυτό. Με τους απλούς συνδέσμους έχουμε το θέμα ότι αν διαγράψει το αρχείο ο ιδιοκτήτης(χρεώνεται μόνο σε αυτόν) και υπάρχουν κα άλλοι που το χρησιμοποιούν τότε το αρχείο δεν διαγράφεται και ο ιδιοκτήτης συνεχίζει να χρεώνεται για αυτό και ας μην το χρησιμοποιεί(το index node παραμένει στον δίσκο).
- **Συμβολικοί σύνδεσμοι:** μόνο μια καταχώρηση σε έναν κατάλογο περιέχει όλες τις πληροφορίες του αρχείου. Το αρχείο σαν οντότητα υπάρχει μια μόνο φορά. Οι υπόλοιποι σύνδεσμοι προς το αρχείο είναι ειδικά αρχεία και μέσα περιέχουν ένα

νέο όνομα διαδρομής που δείχνει προς το κοινόχρηστο αρχείο. Στην ουσία διαβάζουμε μια φορά το symbolic link το οποίο μας ξανακατευθήνει προς το σύστημα αρχείων και ξαναδιαβάζουμε για να βρούμε το πραγματικό αρχείο. Το μειονέκτημα τους είναι ότι δύο διασχίσεις καταλόγων για εντοπισμό αρχείου και επίσης έξτρα index nodes και blocks στον δίσκο.

Το κακό με τους απλούς συνδέσμους είναι ότι αν διαγράψει το αρχείο ο ιδιοκτήτης αυτός θα συνεχίσει να χρεώνεται ακόμα γι'αυτό, ενώ στους συμβολικούς οι σύνδεσμοι απλά θα γίνουν άκυροι αφού το αρχείο υπάρχει μια μόνο φορά.

ΚΑΤΑΓΡΑΦΙΚΑ ΣΥΣΤΗΜΑΤΑ

Οι εγγραφές κοστίζουν παραπάνω από τις αναγνώσεις. Οι αναγνώσεις γίνονται ομαδικά(πολλά μπλοκ μαζί), ενώ οι εγγραφές γίνονται με μικρά κομμάτια και άμεσα όταν τροποποιηθούν(για να μην χάσουμε δεδομένα).

Τα καταγραφικά συστήματα κάνουν το εξής: ότι αλλαγές έχουν γίνει (σε αρχεία, καταλόγους, index nodes) τα εγγράφει όλα μαζί σε μεγάλο συνεχόμενο μπλοκ μνήμης. Όταν το μπλοκ γεμίσει τότε γράφονται όλα στον δίσκο. Το θέμα με αυτό είναι ότι τα index nodes είναι διάσπαρτα ενώ τα θέλουμε όλα μαζεμένα σε ένα σημείο. Η λύση στο θέμα αυτό είναι να έχουμε έναν πίνακα στην μνήμη με δείκτες προς τα index nodes.

Τι γίνεται αν γεμίσει ο δίσκος: επειδή γράφουμε αλλαγές την μια μετά την άλλη στον δίσκο κάποια στιγμή αυτό θα γεμίσει. Μπορεί μια αλλαγή να μην ισχύει πλέον επειδή έχει υπάρξει στην συνέχεια στο ίδιο αρχείο άλλη. Έχουμε ένα **νήμα καθαρισμού** το οποίο σαρώνει διαρκώς το αρχείο από τις πιο παλιές αλλαγές στις πιο καινούριες και άμα δει μια αλλαγή που δεν ισχύει πλέον ελευθερώνει το μπλοκ, ενώ αν δει κάποια ενεργά τότε τα πάει στην αρχή. Ο δίσκος λειτουργεί σαν κυκλική λίστα. Έχουμε πιο περίπλοκη διαχείριση αλλά μεγαλύτερες ταχύτητες εγγραφής.

ΗΜΕΡΟΛΟΓΙΑΚΑ ΣΥΣΤΗΜΑΤΑ

Σε κάθε αλλαγή του συστήματος διατηρούμε ημερολόγιο με τις αλλαγές που πρόκειται να γίνουν. Αυτό το κάνουμε ώστε σε περίπτωση που έχουμε κάποια αποτυχία στο σύστημα π.χ διακοπή ρεύματος να μπορέσουμε να ανακάμψουμε εύκολα και να ολοκληρώσουμε τις αλλαγές που ήταν να γίνουν.

Παράδειγμα προβλήματος με διαγραφή αρχείου: για να διαγράψουμε ένα αρχείο από το σύστημα πρώτα διαγράφουμε το αρχείο από το κατάλογο και μετά αποδεσμεύουμε τον κόμβο i και τέλος αποδεσμεύουμε το μπλοκ του αρχείου. Για να αποφύγουμε αυτό το πρόβλημα γράφουμε στον δίσκο της λειτουργίες που θα εκτελεστούν, εκτελούμε τις λειτουργίες και στην συνέχεια διαγράφουμε την εγγραφή από τον δίσκο, που δηλώνει ότι οι λειτουργίες αυτές εκτελέστηκαν με επιτυχία. Αν τώρα αποτύχει η εγγραφή τότε όταν επανέλθουμε εκτελούμε ξανά τις λειτουργίες που είναι καταγεγραμμένες στο ημερολόγιο.

Οι λειτουργίες αυτές πρέπει να είναι **αδύναμες**, δλδ η επανάληψη τους να μην δημιουργεί πρόβλημα. Π.χ η εγγραφή σε συγκεκριμένη θέση στον δίσκο είναι αδύναμη, ενώ η εγγραφή στο τέλος μιας λίστας δεν είναι αδύναμη λειτουργία.

ΕΙΚΟΝΙΚΑ ΣΥΣΤΗΜΑΤΑ

Ένα τυπικό σύστημα UNIX έχει πολλά συστήματα αρχείων τα οποία λειτουργούν ταυτόχρονα. Το εικονικό σύστημα είναι αυτό που βλέπει ο χρήστης, αλλά από πίσω υπάρχουν πολλά συστήματα. Έχουμε διάκριση της υλοποίησης σε δύο μέροι: 1)κοινό μέρος σε όλα τα συστήματα(διεπαφή), 2)μέρος που εξαρτάται από το σύστημα. Το εικονικό σύστημα λαμβάνει όλες τις κλήσεις του χρήστη και τις μεταβιβάζει στα διάφορα συστήματα. Έχουμε αντικειμενοστραφή υλοποίηση(C). Όταν το σύστημα μας αρχικοποιείται τότε αρχικοποιούνται και όλα τα συστήματα αρχείων.

ΔΙΑΧΕΙΡΙΣΗ ΚΑΙ ΒΕΛΤΙΣΤΟΠΟΙΗΣΗ

ΔΙΑΧΕΙΡΙΣΗ ΧΩΡΟΥ ΔΙΣΚΟΥ

Επιλέγουμε να κάνουμε κατανομή σε μπλοκ(όμοιο της σελιδοποίησης) και το μέγεθος του μπλοκ συνήθως επιλέγουμε να είναι όσο ένας τομές, μια τροχιά, ένας κύλινδρος, μία σελίδα και μπορεί να είναι κάποιο πολλαπλάσιο αυτών. Το μεγάλο μπλοκ δίνει ταχύτερες λειτουργίες, ενώ το μικρό μπλοκ έχει μικρότερη σπατάλη χώρου.

Παρακολούθηση ελεύθερων μπλοκ:

- **Συνδεδεμένη λίστα ελεύθερων μπλοκ:** ένα μπλοκ δεν δείχνει μόνο στο επόμενο του άλλα έχει δείκτες και προς άλλα μπλοκ. Δεν το χρησιμοποιούμε γιατί έχει πολύ αργή διάσχιση των μπλοκ.
- **Χάρτης bit ελεύθερων και δεσμευμένων μπλοκ:** έχουμε τιμή 0 ή 1 για κάθε μπλοκ αναλόγως αν είναι διαθέσιμο ή όχι. Έχει σταθερό μέγεθος ανεξάρτητα το πως χρησιμοποιούμε τον δίσκο αλλά είναι πολύ μεγάλο σε μεγάλους δίσκους.
- **Συνδεδεμένη λίστα ελεύθερων περιοχών:** πολλά συνεχόμενα ελεύθερα μπλοκ αποτελούν μια περιοχή. Οπότε αυτό που κάνουμε είναι να αποθηκεύσουμε το πρώτο μπλοκ και ένα μετρητή για το πλήθος των μπλοκ. Αποφεύγουμε να χρησιμοποιούμε αυτή τη μέθοδο διότι αν οι περιοχές αποτελούνται από ένα μπλοκ είναι πολύ σπάταλη.

Συνήθως χρησιμοποιούμε μια βελτιστοποίηση της συνδεδεμένης λίστας ελεύθερων μπλοκ.

Μόνο ένα μπλοκ της λίστας βρίσκεται στην μνήμη, τα υπόλοιπα είναι στον δίσκο. Αν τώρα έχουμε χώρο για δύο ακόμα μπλοκ, αυτό που κάνουμε όταν γεμίζει το μπλοκ στην μνήμη είναι να το σπάσουμε σε δύο μισά και το ένα να μείνει στην μνήμη και το άλλο στον δίσκο. Παρόμοια τεχνική έχουμε και για τους χάρτες bit, ένα block βρίσκεται στην μνήμη.

ΑΝΤΙΓΡΑΦΑ ΑΣΦΑΛΕΙΑΣ

Το σύστημα αρχείων είναι πολύ κρίσιμο γιατί αν καταστραφεί ο δίσκος τότε μπορεί να χάσουμε χρόνια δουλειάς και πολύ σημαντικές πληροφορίες. Η αντιγραφή των αρχείων πρέπει να γίνει σε χώρο μακριά από τον δίσκο και συνήθως γίνεται σε ταινία, CD's ή κάποια άλλα μέσα. Τα αντίγραφα χρειάζονται για ανάκαμψη από καταστροφές όπως φωτιά στον δίσκο, αλλά και από ανάκαμψη από σφάλματα χρήστη.

Δεν είναι απαραίτητο να έχουμε εφεδρικά αντίγραφα για όλα τα αρχεία, αυτά που χρειάζεται να αντιγράψουμε είναι τα αρχεία του χρήστη.

Πότε πρέπει να αντιγράφονται τα αρχεία: γενικά δεν τροποποιούνται υπερβολικά πολλά αρχεία σε ένα σχετικά μικρό χρονικό διάστημα κάποιων ημερών. Χρησιμοποιούμε την τεχνική των **αυξητικών αντιγράφων**. Αυτό που γίνεται είναι περιοδικά να παίρνουμε ένα πλήρες αντίγραφο των αλλαγών που έγιναν και να αντιγράψουμε αυτά τα οποία άλλαξαν σε σχέση με την τελευταία εγγραφή. Μπορούμε και στο μεσοδιάστημα να αντιγράψουμε αλλαγμένα αρχεία.

Για λόγους ασφαλείας είναι καλύτερο να μην συμπιέζουμε τα αρχεία ώστε να μην χάσουμε τα δεδομένα μας αν αλλοιωθεί ένα μέρος του αντιγράφου.

Τα ενεργά αρχεία αντιγράφονται με χρήση αλγορίθμων για δημιουργία στιγμιότυπων και καταλήγουμε είτε με την κατάσταση που είχε το αρχείο πριν τις αλλαγές είτε με την κατάσταση του μετά τις αλλαγές.

Φυσική αντιγραφή δίσκου: γράφοντα όλα τα μπλοκ μιας διαμέρισης με τη σειρά και παραλείπουμε τα ελεύθερα μπλοκ σημειώνοντας όμως την ύπαρξη τους. Η διαδικασία αυτή είναι γρήγορη και απλή.

Λογική αντιγραφή δίσκου: κάνουμε μια αναδρομική διάσχιση των αρχείων και σημειώνουμε ποια θα αντιγράψουμε. Χρήση χάρτη bit για τους κόμβους i.

Επαναφορά αρχείων: ξεκινάμε από το τελευταίο πλήρες αντίγραφο και το επαναφέρουμε.

ΣΥΝΕΠΕΙΑ ΣΥΣΤΗΜΑΤΟΣ ΑΡΧΕΙΩΝ

Έστω πως αποτυγχάνουμε στη μέση μιας λειτουργίας π.χ όταν δημιουργούμε ένα αρχείο, τα μπλοκ που είχαν δεσμευτεί για αυτό δεν χρησιμοποιούνται αλλά δεν φαίνονται ελεύθερα και βρίσκονται στην λίστα καταλόγων. Για να

αποφύγουμε τέτοιες ασυνέπειες χρησιμοποιούμε ειδικά προγράμματα ελέγχου συνέπειας. Αυτό που κάνουμε είναι να έχουμε δύο πίνακες μεγέθους όσο και τα μπλοκ. Ο πρώτος δείχνει το πόσες φορές υπάρχει το αρχείο αυτό, που είναι γραμμένο στο χ μπλοκ, σε καταλόγους και ο δεύτερος δείχνει αν το μπλοκ είναι ελεύθερο ή όχι. Έχουμε τις εξής περιπτώσεις σφαλμάτων:

I. Να έχουμε τιμή 0 και στους δύο πίνακες, αυτό που κάνουμε είναι να δηλώσουμε το μπλοκ ως ελεύθερο και να κάνουμε την τιμή του στα ελεύθερα 1.
II. Να έχουμε τιμή παραπάνω από 1 σε μπλοκ ελεύθερο στην μνήμη, το κάνουμε 1.
III. Να έχουμε τιμή μεγαλύτερη του 1 στα χρησιμοποιούμενα. Στην περίπτωση των κόμβων ι κάθε κόμβος δείχνει σε διαφορετικό αρχείο οπότε πρέπει να κάνουμε την τιμή του 1.

ΕΠΙΔΟΣΗΣ ΣΥΣΤΗΜΑΤΟΣ ΑΡΧΕΙΩΝ

Για να επιταχύνουμε διαδικασίες ανάγνωσης έχουμε μια κρυφή μνήμη(block cache) οι οποία έχει εκεί τα μπλοκ που έχουν διαβαστεί πιο πρόσφατα και αν χρειαστεί να τα ξαναδιαβάσουμε τότε καταφεύγουμε στην μνήμη και όχι στον δίσκο. Αν διαβάσουμε ένα μπλοκ από τον δίσκο θα πρέπει να αφαιρέσουμε ένα μπλοκ. Θα μπορούσαμε να κάνουμε LRU, αλλά κάνουμε μια μικρή τροποποίηση και εκτιμούμε αν θα χρειαστεί ξανά το μπλοκ και αναλόγως το φέρνουμε στην αρχή της λίστας στο cache block ή στο τέλος.

Προσπαθούμε μπλοκ που ανήκουν στο ίδιο αρχείο να είναι γραμμένα συνεχόμενα στον δίσκο, αυτό όμως δεν είναι εφικτό οπότε προσπαθούμε να είναι σε κοντινές περιοχές ώστε να μην μετακινείται η κεφαλή του δίσκου πολύ.

ΑΝΑΣΥΓΚΡΟΤΗΣΗ ΔΙΣΚΩΝ

Στον δίσκο όταν δημιουργούνται κενά από διαγραφή αρχείων αυτό που κάνουμε είναι ανασυγκρότηση, ομαδοποιούμε τα μπλοκ και τα γράφουμε συνεχόμενα σε αυτά τα τμήματα μπλοκ.

ΠΑΡΑΔΕΙΓΜΑΤΑ ΣΥΣΤΗΜΑΤΩΝ ΑΡΧΕΙΩΝ

ΣΥΣΤΗΜΑΤΑ ΑΡΧΕΙΩΝ CD-ROM

Χρησιμοποιούνται για μέσα που γράφονται μια μόνο φορά

ΣΥΣΤΗΜΑΤΑ ΑΡΧΕΙΩΝ MS-DOS

Έχουμε μεταβλητό πλήθος καταχωρήσεων ανά κατάλογο, αλλά σταθερό μέγεθος καταχώρησης. Χρησιμοποιούμε FAT που δείχνει σε συστοιχίες των 521 byte.

ΣΥΣΤΗΜΑΤΑ ΑΡΧΕΙΩΝ UNIX V7

Βασική μορφή δέντρου, ακυκλικός γράφος με χρήση συνδέσμων. Χρήση κόμβων ι.

ΕΝΟΤΗΤΑ 6: ΑΔΙΕΞΟΔΑ

ΠΟΡΟΙ

ΤΥΠΟΙ ΠΟΡΩΝ

Έχουμε πόρους υλικού(εκτυπωτής) και πόρους λογισμικού(πίνακας διεργασιών). Ορισμένοι πόροι του ΛΣ δεν είναι κοινόχρηστοι. Όταν πάμε να δεσμεύσουμε πόρους μπορούμε να οδηγηθούμε σε αδιέξοδο. Παράδειγμα αποτελεί η δέσμευση του εκτυπωτή και του σαρωτή από δυο διεργασίες με διαφορετική σειρά. Η μια θα έχει τον εκτυπωτή, η άλλη τον σαρωτή και θα περιμένουν και οι δύο να τελειώσει η άλλη και θα οδηγηθούμε σε αδιέξοδο. Τα αδιέξοδα είναι γενικό πρόβλημα και μπορούν να συμβούν με πολλές διεργασίες(κυκλική αναμονή) ή με βάσεις(κλείδωμα εγγραφών).

- **Προεκτοπίσιμοι πόροι:** μπορούν να αποδεσμευτούν εύκολα, όπως π.χ η μνήμη.
- **Μη προεκτοπίσιμοι πόροι:** αποδεσμεύονται δύσκολα και αν το κάνουμε αυτό θα έχουμε μεγάλο κόστος, π.χ εγγραφέας CD.

Τα αδιέξοδα αφορούν μη προεκτοπίσιμους πόρους. Οι πόροι αυτοί δεσμεύονται με κλήση συστήματος request ή με άνοιγμα αρχείου συσκευής.

Όταν ένας πόρος δεν είναι διαθέσιμος η διεργασία είτε μπλοκάρεται είτε στέλνει μήνυμα σφάλματος και ξαναπροσπαθεί.

ΑΠΟΚΤΗΣΗ ΠΟΡΩΝ

Δέσμευση επιπέδου χρήστη(π.χ βάσεις): χρήση δυαδικού σηματοφόρου με αρχική τιμή 1.

Αποδέσμευση αντίστροφα από τη δέσμευση.

ΑΔΙΕΞΟΔΑ

ΟΡΙΣΜΟΣ ΑΔΙΕΞΟΔΟΥ

Τυπικός ορισμός αδιέξοδου: έστω ότι έχουμε κάποιες διεργασίες οι οποίες περιμένουν κάποιο συμβάν και το συμβάν αυτό παράγεται μόνο εσωτερικά. Και αφού όλες οι διεργασίες περιμένουν συμβάν από κάποια άλλη του συνόλου τότε καμία δεν θα προχωρήσει. Κάνουμε τις υποθέσεις ότι έχουμε ένα νήμα ανά διεργασία και ότι οι διεργασίες δεν αφυπνίζονται από διακοπές.

Αδιέξοδα πόρων: αυτό που συμβαίνει είναι ότι οι πόροι κατέχονται ένας ανά διεργασία. Έστω τώρα ότι η διεργασία Α περιμένει πόρο από την Β, η Β από την Γ, η Γ από την Α για να προχωρήσουν, τότε έχουμε μια κυκλική ουρά αναμονής και άρα αδιέξοδο.

ΣΥΝΘΗΚΕΣ ΤΟΥ COFFMAN

Σε ένα σύστημα πρέπει να ισχύουν όλες οι συνθήκες Coffman για να έχουμε αδιέξοδο, αν έστω και μία δεν ισχύει τότε δεν έχουμε αδιέξοδο.

- **Αμοιβαίος αποκλεισμός:** κάθε πόρος είτε είναι εκχωρημένος σε μια διεργασία είτε είναι διαθέσιμος. Δεν επιτρέπεται καταμερισμός των πόρων.
- **Δέσμευση και αναμονή:** μια διεργασία με εκχωρημένους πόρους μπορεί να ζητήσει και άλλους. Δεν ζητάει υποχρεωτικά όλους τους πόρους μαζί.
- **Μη προεκτόπιση:** δεν μπορούμε να αφαιρέσουμε ήδη εκχωρημένους πόρους.
- **Κυκλική αναμονή:** κυκλική αλυσίδα διεργασιών που περιμένουν η μια την άλλη.

ΜΟΝΤΕΛΟΠΟΙΗΣΗ ΑΔΙΕΞΟΔΩΝ

Μοντελοποίηση αδιεξόδων με γράφους: οι διεργασίες συμβολίζονται με κύκλους και οι πόροι με τετράγωνα. Αν έχουμε βέλος από διεργασίας προς πόρο σημαίνει ότι η διεργασία ζητάει τον πόρο, ενώ α έχουμε βέλος από πόρο προς διεργασία σημαίνει ότι η διεργασία κατέχει τον πόρο. Αν έχουμε κύκλο στον γράφο τότε έχουμε αδιέξοδο.

ΑΝΤΙΜΕΤΩΠΙΣΗ ΑΔΙΕΞΟΔΩΝ

Τέσσερις γενικές στρατηγικές αντιμετώπισης:

- **Αγνόηση προβλήματος:** ελπίζουμε να μην συμβούν.
- **Εντόπιση και ανάκαμψη:** απαιτεί ακύρωση ενεργειών και σπατάλη πόρων.
- **Αποφυγή:** όταν υπάρχει απειλή εμφάνισης αδιεξόδων να μην επιτρέπουμε κάποιες ενέργειες. Απαιτεί προσεκτική κατανομή πόρων.
- **Αποτροπή:** άρση κάποιας από τις συνθήκες του Coffman και έτσι θα είναι αδύνατον να συμβεί αδιέξοδο.

ΑΓΝΟΗΣΗ ΑΔΙΕΞΟΔΩΝ

Ο αλγόριθμος της στρουθοκαμήλου: απλά αγνοούμε το πρόβλημα, θεωρητικά η προσέγγιση αυτή είναι απαράδεκτη, αλλά στην πραγματικότητα αν σκεφτούμε ότι τα αδιέξοδα δεν συμβαίνουν πολύ συχνά και η αντιμετώπιση τους έχει κάποιο κόστος τότε αυτό αποτελεί μια λύση. Σε συστήματα όπως ΛΣ των PC που έχουμε αρκετά συχνά επανεκκίνηση και αν συμβεί ένα αδιέξοδο τότε αυτό θα πάψει να υπάρχει στην επόμενη επανεκκίνηση οπότε δεν υπάρχει κάποιο πολύ σοβαρός λόγος να ασχοληθούμε με αυτά.

Αυτό που συνήθως γίνεται από τα περισσότερα ΛΣ είναι να εφαρμόσουν, εν μέρει, κάποιες από τις πολιτικές αντιμετώπισης αδιεξόδων, για να περιορίσουν τον κίνδυνο τους.

ΕΝΤΟΠΙΣΜΟΣ ΚΑΙ ΑΝΑΚΑΜΨΗ

ΜΕ ΕΝΑΝ ΠΟΡΟ ΑΝΑ ΕΙΔΟΣ

Αλγόριθμος για εύρεση αδιεξόδου

Για κάθε κόμβο Κ εκτελούμε τα επόμενα βήματα από τον Κ:

1. Αρχικά έχουμε την λίστα Λ που είναι κενή και όλα τα τόξα είναι ασημείωτα.
2. Προσθέτουμε τον τρέχοντα κόμβο στο τέλος της Λ.
3. Αν ο κόμβος εμφανίζεται δυο φορές στη Λ, η Λ περιέχει κύκλο.
4. Αν δεν υπάρχουν ασημείωτα τόξα από τον κόμβο τότε πάμε στο 7.
5. Επιλέγουμε και σημειώνουμε ένα ασημείωτο τόξο.
6. Χρησιμοποιούμε τον νέο κόμβο ως τρέχοντα και πάμε στο 2.
7. Αν ο τρέχων κόμβος είναι ο αρχικός, τέλος χωρίς κύκλο.
8. Αλλιώς αφαιρούμε τον κόμβο από τη λίστα.
9. Κάνουμε τον προηγούμενο της λίστας τρέχοντα και πάμε στο 2.

Αυτό που πραγματικά κάνουμε είναι μια διάσχιση κατά βάθος για να δούμε αν υπάρχει κύκλος. Κάνουμε DFS.

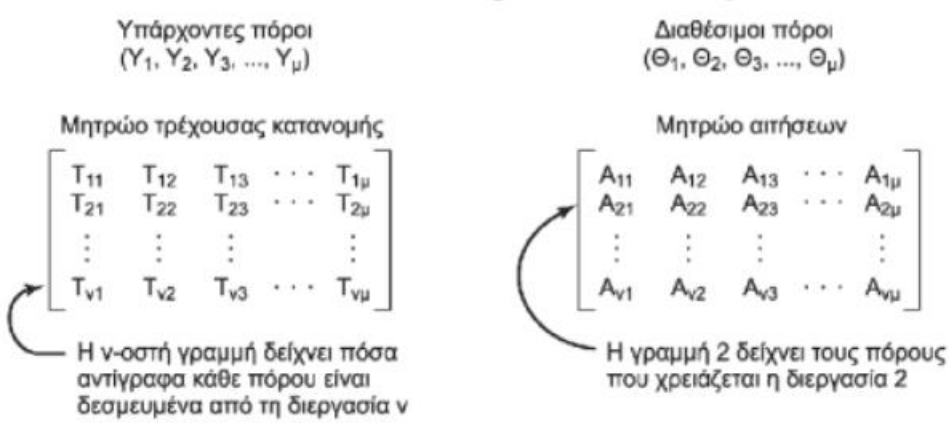
ΜΕ ΠΟΛΛΟΥΣ ΠΟΡΟΥΣ ΑΝΑ ΕΙΔΟΣ

Εντοπισμός αδιεξόδων με πολλούς πόρους ανά είδος.

Έχουμε δύο διανύσματα **Υ** και **Θ**. Το διάνυσμα Υ δείχνει τους πόρους που υπάρχουν ανά είδος και το διάνυσμα Θ δείχνει πόσοι είναι διαθέσιμοι.

Έχουμε και δύο πίνακες **Τ** και **Α**. Το μητρώο κατανομής Τ δείχνει που έχουν εκχωρηθεί οι πόροι και το μητρώο αιτήσεων Α πόσοι μπορεί να ζητηθούν.

Η οριζόντια γραμμή των πινάκων έχει θέσεις όσο και οι διαφορετικοί πόροι και η κάθετη έχει θέσεις όσο και οι διεργασίες.



Βασική συνθήκη πόρων: το άθροισμα των κατανεμημένων και των διαθέσιμων πρέπει να είναι ίσο με το πλήθος των υπαρκτών.

Υποθέσεις αλγορίθμου εντοπισμού αδιεξόδων:

- Για δύο διανύσματα Α και Β για να ισχύει Α<=Β θα πρέπει όλα τα στοιχεία του Α να είναι <= των αντίστοιχων του Β.
- Κάθε διεργασία διατηρεί τους πόρους μέχρι τέλους.
- Αρχικά όλες οι διεργασίες είναι μη σημειωμένες.
- Όσες παραμείνουν ασημείωτες βρίσκονται σε αδιέξοδο.

Αλγόριθμος εντοπισμού αδιεξόδων

1. Ψάξε μια ασημείωτη διεργασία κ με Ακ<=Θ, δλδ οι πόροι που ζητάει είναι διαθέσιμοι και μπορούν να της δοθούν.
2. Αν βρεθεί τέτοια, πρόσθεσε το Τκ στο Θ, δλδ επέστρεψε τους πόρους που είχε δεσμεύσει πριν.
3. Μετά σημείωσε τη διεργασία και πήγαινε στο 1.
4. Αν δεν υπάρχει, ο αλγόριθμος τερματίζει.

Αυτό που κάνουμε είναι να ψάξουμε διεργασία που μπορεί να ολοκληρωθεί. Αν όλες οι διεργασίες στο τέλος είναι σημειωμένες σημαίνει ότι υπάρχει ένας τρόπος εκτέλεσης των διεργασιών που δεν οδηγεί σε αδιέξοδο. Αν τώρα δεν είναι όλες σημειωμένες τότε υπάρχει περίπτωση να οδηγηθούμε σε αδιέξοδο ή όχι ανάλογα το πώς θα γίνει η εκτέλεση. Αν είναι όλες ασημείωτες τότε έχουμε αδιέξοδο.

ΑΝΑΚΑΜΨΗ ΑΠΟ ΑΔΙΕΞΟΔΑ

Ο παραπάνω αλγόριθμος εκτελείται σε στατική κατάσταση. Τον εκτελούμε όταν θέλουμε να ελέγχουμε για αδιέξοδα. Ο έλεγχος θα μπορούσε να γίνει όποτε ζητάμε κάποιον πόρο αλλά αυτό έχει κόστος. Συνήθως επιλέγουμε να κάνουμε έλεγχο είτε περιοδικά(κάθε κ χτύπους ρολογιού) είτε όταν έχει πέσει ο φόρτος πολύ(μπορεί να οφείλεται σε αδιέξοδο).

Ανάκαμψη μέσω προεκτόπισης(δλδ να πάρουμε κάποιους πόρους): έχουμε ως παράδειγμα ένα αδιέξοδο που εμπλέκει τον εκτυπωτή. Διακόπτουμε την διεργασία που έχει τον εκτυπωτή, μαζεύουμε την έξοδο της και παραχωρούμε τον εκτυπωτή. Όταν τελειώσει ενεργοποιούμε ξανά την αρχική διεργασία. Αυτή η μέθοδος δεν είναι καλή και αρκετές φορές δε είναι εφικτή, παράδειγμα αποτελεί ένας εγγραφέας CD που αν διακοπεί τότε τερματίζεται και πρέπει από την αρχή να ξανακάνουμε record.

Ανάκαμψη μέσω ανακατασκευής(rollback): περιοδικά δημιουργούμε σημεία ελέγχου(checkpoints) και κρατάμε κρατάμε τις καταστάσεις των διεργασιών σε αρχεία. Αυτό που κάνουμε είναι να εντοπίσουμε τον πόρο που έχει πρόβλημα, την διεργασία που τον χρησιμοποιεί και να γυρίσουμε σε ένα σημείο όπου η διεργασία δεν είχε δεσμεύσει τον πόρο. Αυτό έχει κόστος, τα checkpoints περιλαμβάνουν εικόνα μνήμης και πόρους,

Ανάκαμψη μέσω εξάλειψης διεργασιών:

- Σκοτώνουμε μια διεργασία του κύκλου: ελπίζουμε να σπάσει ο κύκλος.
- Σκοτώνουμε μια διεργασία με κατάλληλους πόρους: αυτούς που ζητούνται από τον κύκλο.

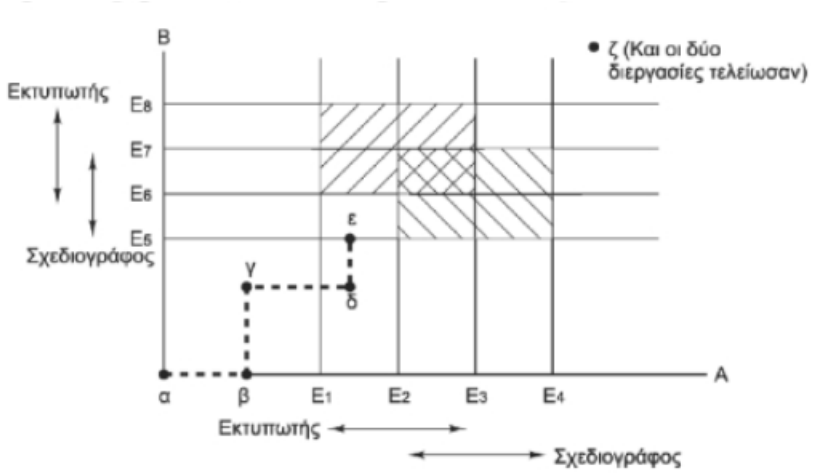
Στόχος είναι να σπάσει ο κύκλος. Προτιμάμε να σκοτώνουμε διεργασίες που τερματίζονται εύκολα και δεν θα προκαλέσουμε θέμα στο σύστημα στο μέλλον. Μια κατάλληλη διεργασία είναι η μεταγλώττιση που μπορούμε απλά να την ξεκινήσουμε από την αρχή και μια ακατάλληλη είναι η βάση δεδομένων η οποία μπορεί να βρίσκεται σε ασυνεπή κατάσταση και να χάσουμε σημαντικές πληροφορίες.

ΑΠΟΦΥΓΗ ΑΔΙΕΞΟΔΩΝ

Η λύση με εντοπισμό αδιεξόδων δεν τόσο ρεαλιστική και απαιτεί γνώσεις που σπανίως έχουμε(απαιτήσεις πόρων) και είναι πολύ συντηρητική(υποθέτει ότι η διεργασία θα ζητήσει όλους τους πόρους μαζί). Μια πιο ρεαλιστική λύση είναι η αποφυγή αδιεξόδων η οποία ασχολείται με το τι πραγματικά κάνει η διεργασία και όχι με τις μέγιστες απαιτήσεις. Έτσι διακόπτουμε ή μπλοκάρουμε μια διεργασία όταν υπάρχει πραγματικά απειλή αδιεξόδου αυτή την στιγμή και όχι όταν υπάρχει απλά η πιθανότητα για αδιέξοδο.

ΤΡΟΧΕΙΣ ΠΟΡΩΝ

Έστω πως έχουμε δυο διεργασίες και δύο πόρους, κάθε διεργασία χρειάζεται τους πόρους για κάποιο διάστημα. Έχουμε τους άξονες(κάθετο και οριζόντιο) που δείχνουν την εξέλιξη μιας διεργασίας στον χρόνο και έχουμε μια τεθλασμένη γραμμή που δείχνει την εξέλιξη των διεργασιών στον χρόνο και μπορεί να κινηθεί μόνο αριστερά και δεξιά.



Πρέπει να αποφύγουμε τις σκιασμένες περιοχές και αυτό μπορεί να γίνει αν δεν δώσουμε ην χρονική στιγμή **ε** τον σχεδιογράφο στην Β και συνεχίσουμε την Α.

ΑΣΦΑΛΕΙΣ ΚΑΤΑΣΤΑΣΕΙΣ

Ασφαλείς και ανασφαλής καταστάσεις

Έχουμε ίδια κατάσταση με τον αλγόριθμο εντοπισμού, δλδ έχουμε τα διανύσματα Υ και Θ και τους πίνακες Τ και Α.

Μια κατάσταση είναι ασφαλής αν δεν είμαστε ήδη σε αδιέξοδο και υπάρχει κάποια σειρά ολοκλήρωσης των διεργασιών.

Αν είμαστε σε ανασφαλή κατάσταση δεν σημαίνει απαραίτητα ότι θα έχουμε αδιέξοδο, αλλά ούτε είναι εγγυημένο ότι δεν θα έχουμε.

ΑΛΓΟΡΙΘΜΟΣ ΤΡΑΠΕΖΙΤΗ

Ο αλγόριθμος του τραπεζίτη για έναν πόρο: η ιδέα βασίζεται στον αλγόριθμο εντοπισμού του Dijkstra και είναι μια επέκταση του αλγορίθμου του εντοπισμού αδιεξόδων. Πριν δώσουμε κάποιον πόρο βλέπουμε αν είμαστε σε ασφαλή κατάσταση ή όχι, αν δεν είμαστε τότε δεν παραχωρούμε τον πόρο.

Ο αλγόριθμος του τραπεζίτη για πολλούς πόρους: έχουμε Υ, Θ, Τ, Α.

1. Βρες μια διεργασία R που μπορεί να ολοκληρωθεί. Η γραμμή της R στο Α πρέπει να είναι <=Θ. Δεσμεύονται οι πόροι και μετά τερματίζει η R.
2. Απελευθέρωσε τους πόρους προσθέτοντας τους στο Θ.
3. Επανέλαβε τα βήματα 1 και 2.
    - Αν τερματίζουν όλες οι διεργασίες τότε είσαι σε ασφαλή κατάσταση, αλλιώς σε ανασφαλή.

ΑΠΟΤΡΟΠΗ ΑΔΙΕΞΟΔΩΝ

Η αποφυγή αδιεξόδων είναι λίγο καλύτερη λύση από τον εντοπισμό. Όμως είναι πολύ συντηρητικός και βασίζεται και αυτός στις μέγιστες απαιτήσεις διεργασιών που δεν είναι γνωστές.

Η λύση στην αποτροπή αδιεξόδων είναι η προσβολή των συνθηκών του Coffman, αρκεί μια να μην ισχύει και τότε δεν έχουμε αδιέξοδα. Αυτό όμως απαιτεί επιβολή περιορισμών στο σύστημα.

- **Προσβολή συνθήκης αμοιβαίου αποκλεισμού:** ορισμένοι πόροι όπως ο εκτυπωτής απαιτούν αμοιβαίο αποκλεισμό. Αυτό που γίνεται είναι να κρύψουμε τον εκτυπωτή πίσω από μια διεργασία/διαχειριστή και μόνο αυτή να μπορεί να τυπώνει. Έτσι δεν θα υπάρξει ποτέ αδιέξοδο στον εκτυπωτή. Η διεργασία θα έχει μια ουρά εκτύπωσης και όσες διεργασίες θέλουν να εκτυπώσουν θα περιμένουν εκεί. Υπάρχει ο κίνδυνος να έχουμε αδιέξοδο στην ουρά εκτύπωσης, αλλά είναι πολύ πιο μικρός, γιατί η ουρά θα δεσμευτεί για πολύ μικρό χρόνο όσο ενημερωθεί με την εισαγωγή νέας αίτησης, ενώ ο εκτυπωτής δεσμεύεται για πολύ μεγαλύτερο χρόνο.
- **Προσβολή συνθήκης δέσμευσης και αναμονής:** το σκεπτικό εδώ είναι να δεσμεύσουμε όλους τους πόρους από την αρχή και αν δεν μας επαρκούν να δοκιμάσουμε πιο μετά. Αυτό όμως είναι τόσο κακό όσο και η αποφυγή οπότε δεν το κάνουμε.
- **Προσβολή συνθήκης μη προεκτόπισης:** σε ορισμένες συσκευές όπως ο εκτυπωτής μη εφικτό(χάνουμε χαρτί…). Μπορούμε να το παραβιάσουμε με ίδιο τρόπο με τον αμοιβαίο αποκλεισμό, δλδ να έχουμε έναν διαχειριστή πόρων.
- **Προσβολή συνθήκης κυκλικής αναμονής:** αριθμούμε όλους τους πόρους του συστήματος. Οι διεργασίες ζητούν πόρους μόνο με αύξουσα σειρά, δλδ αν έχεις τον πόρο 2 δεν μπορείς να ζητήσεις τον 1. Έτσι αποφεύγουμε τους κύκλους, αλλά περιορίζουμε τον τρόπο λειτουργίας των διεργασιών. Μια παραλλαγή αυτού είναι να ζητάμε σίγουρα αύξοντες πόρους αλλά σε σχέση με το τι έχουμε δεσμεύσει και ζητήσει αυτή τη στιγμή, δλδ αν πριν είχαμε ζητήσει τον πόρο 4 αλλά τον αποδεσμεύσαμε μπορούμε τώρα να ζητήσουμε τον 3. Το μόνο πρόβλημα με την λύση αυτή είναι να βρούμε τη σωστή αρίθμηση πόρων που δεν είναι καθόλου προφανής.

Συμπέρασμα: από την προσβολή των τεσσάρων συνθηκών του Coffman μόνο οι δύο λειτουργούν καλά:

- ✓ Η προσβολή του αμοιβαίου αποκλεισμού και
- ✓ Η προσβολή της κυκλικής αναμονής.

ΑΛΛΑ ΘΕΜΑΤΑ

ΚΛΕΙΔΩΜΑ ΣΕ ΔΥΟ ΦΑΣΕΙΣ

Χρησιμοποιείται σε βάσεις δεομένων. Αυτό που κάνουμε είναι να προσπαθήσουμε αρχικά να κλειδώσουμε τις εγγραφές, αν καταφέρουμε και τις κλειδώσουμε όλες τότε προχωράμε στην δεύτερη φάση, αν δεν τα καταφέρουμε ξαναπροσπαθούμε. Στην δεύτερη φάση ενημερώνουμε τις εγγραφές και μετά τις ξεκλειδώνουμε. Αυτό είναι παρόμοια με το τι κάνουμε στην αποφυγή, με τη διαφορά ότι στην αποφυγή καθορίζουμε τις μέγιστες απαιτήσεις από την αρχή, ενώ στο κλείδωμα κατά τη διάρκεια τις διεργασίας και όχι για πολύ μεγάλο χρονικό διάστημα.

ΑΔΙΕΞΟΔΑ ΕΠΙΚΟΙΝΩΝΙΑΣ

Έστω η διεργασία Α στέλνει μήνυμα στην Β. Η απάντηση που έστειλε η Β χάνεται και η Α μπλοκάρεται για πάντα. Η Β περιμένει και αυτή απάντηση από την Α αφού η ίδια απάντησε, οπότε μπλοκάρεται και αυτή.

Η λύση είναι τα χρονόμετρα. Αν περάσει κάποιος χρόνος και δεν έχουμε λάβει απάντηση ξαναστέλνουμε μήνυμα. Η ερώτηση εδώ είναι τι θα γίνει αν η απάντηση απλά άργησε να μας έρθει.

ΕΝΕΡΓΟ ΑΔΙΕΞΟΔΟ

Ενεργό αδιέξοδο σε κρίσιμες περιοχές. Έστω ότι χρησιμοποιούμε αναμονή με απασχόληση. Δύο διεργασίες προσπαθούν να αποκτήσουν πόρους με διαφορετική σειρά, τότε όταν αποκτούν τον έναν πόρο θα συνεχίσουν να προσπαθούν αν αποκτήσουν τον άλλον ανεπιτυχώς και δεν θα κάνουν καμία πρόοδο. Η κατάσταση αυτή, δλδ όταν οι διεργασίες εκτελούνται αλλά δεν κάνουν πρόοδο λέγεται ενεργό αδιέξοδο.(livelock).

ΛΙΜΟΚΤΟΝΙΑ

Η αναβολή εξυπηρέτησης μιας διεργασίας λέγεται λιμοκτονία. Δεν είναι αδιέξοδος, αλλά κάποιες διεργασίες μπορεί να περιμένουν για πάντα. Μπορούμε να το αποφύγουμε αυτό με κατάλληλους αλγορίθμους χρονοπρογραμματισμού. Δεν προτιμάμε τον FCFS(first come first served) ο οποίος βέβαια εγγυημένα δεν οδηγεί σε λιμοκτονία. Προτιμάμε αλγορίθμους σαν το RR(round robin).

ΕΝΟΤΗΤΑ 7: ΑΣΦΑΛΕΙΑ

ΠΕΡΙΒΑΛΛΟΝ ΑΣΦΑΛΕΙΑΣ

ΕΙΣΑΓΩΓΗ

Το πρόβλημα της ασφάλειας: ο χρήστης σε μια συσκευή αποθηκεύει εμπιστευτικές πληροφορίες που δεν θέλει να διαρρεύσουν ως προς τρίτους. Η ασφάλεια των πληροφοριών αυτών δυσκολεύεται εξαιτίας των πολύπλοκων ΛΣ και της δικτύωσης. Αφού πλέον υπάρχει η πρόσβαση στο διαδίκτυο μπορούμε να δεχθούμε επιθέσεις από παντού, από άτομα από οποιαδήποτε περιοχή του πλανήτη, ενώ πιο παλιά όταν δεν υπήρχε το διαδίκτυο οι επιθέσεις μπορούσαν να γίνουν από χρήστες που χρησιμοποιούσαν την ίδια μηχανή.

ΤΟ ΠΕΡΙΒΒΑΛΟΝ ΑΣΦΑΛΕΙΑΣ

Η ασφάλεια έχει πολλές όψεις:

-Λειτουργικό σύστημα

-Δίκτυο υπολογιστών

-Πληροφοριακό σύστημα(π.χ μια βάση δεδομένων με πολλούς χρήστες)

Ορολογία

Ασφάλεια: τα δεδομένα δεν θα διαβαστούν ή τροποποιηθούν.

Μηχανισμοί προστασίας: μέθοδοι επίτευξης των στόχων της ασφάλειας.

ΑΠΕΙΛΕΣ

Κάθε σύστημα έχει κάποιους στόχους ασφάλειας. Κάθε στόχος αντιμετωπίζει απειλές.

- **Εμπιστευτικότητα δεδομένων(confidentiality):** απόκρυψη μυστικών δεδομένων. Ο ιδιοκτήτης θα καθορίσει ποιος μπορεί να τα δει.
- **Ακεραιότητα(Integrity):** η τροποποίηση των δεδομένων να μπορεί να γίνει μόνο με άδεια του χρήστη. Να μην μπορούν να διαγραφούν, προστεθούν ή αλλαχθούν τα δεδομένα. Η τροποποίηση των δεδομένων μπορεί να γίνει και χωρίς να μπορούμε να διαβάσουμε τα αρχεία, αλλά συνήθως αν παραβιάζεται η ακεραιότητα τότε παραβιάζεται και η εμπιστευτικότητα.
- **Διαθεσιμότητα συστήματος(availability):** άλλοι χρήστες μπορούν να βαρύνουν το λειτουργικό μας με κάποιο φόρτο ώστε να μην μπορεί να ασχολείται με τις εργασίες μας και έτσι αυτό αχρηστεύει. Συνήθως υπάρχουν επιθέσεις άρνησης

εξυπηρέτησης που στοχεύουν κυρίως σε εξυπηρετητές και άρα επηρεάζουν πολλούς χρήστες μαζί και όχι μεμονωμένα. Οι επιθέσεις αυτές συνήθως γίνονται εναντίον ομάδων και στοχεύουν σε οργανισμούς.

- **Αποκλεισμός εισβολέων:** θέλουμε να αποφύγουμε την κατάληψη του συστήματος μας. Υπάρχουν χρήστες που προσπαθούν να μετατρέψουν το σύστημα μας σε ζόμπι ώστε να έχουν τον πλήρη έλεγχο. Παράδειγμα αποτελεί η ένταξη του συστήματος σε ένα spam botnet το οποίο για να καταφέρει να στείλει χιλιάδες μηνύματα καταλαμβάνει πάρα πολλούς υπολογιστές, πολλές φορές χωρίς οι χρήστες να έχουν καταλάβει κάτι.

ΕΙΣΒΟΛΕΙΣ

Εισβολείς ή εχθροί:

- **Παθητικοί:** διαβάζουν δεδομένα
- **Ενεργητικοί:** τροποποιούν δεδομένα

Το έργο των παθητικών εισβολέων είναι πιο απλό γιατί απλά διαβάζουν, ενώ οι ενεργητικοί τροποποιώντας τα δεδομένα πρέπει να καλύψουν τα ίχνη τους για να μην τους καταλάβουν.

Έχουμε κάποια είδη εισβολέων:

1. **Εσωτερικοί εισβολείς:**
    - **Απλοί χρήστες χωρίς τεχνικές γνώσεις:** διαβάζουν απροστάτευτα δεδομένα και εκμεταλλεύονται κάποια αδυναμία του ΛΣ στο θέμα ασφάλειας ή κακή προστασία των δεομένων από τον ιδιοκτήτη. Κίνητρο τους είναι απλά η περιέργεια.
    - **Εσωτερικοί χρήστες με τεχνικές γνώσεις:** έχουν πρόσβαση στο σύστημα και συνήθως είναι διαχειριστές, φοιτητές, προγραμματιστές συστήματος. Αυτοί αντιμετωπίζουν την προστασία σαν πρόκληση.
2. **Εξωτερικοί εισβολείς:**
    - **Επαγγελματίες με στόχο το κέρδος:** εκμεταλλεύονται κενά του συστήματος και συνήθως πληρώνονται από κάποιον για να εισβάλουν στο σύστημα.
    - **Εμπορικοί και στρατιωτικοί:** στόχος να κλέψουν δεδομένα ανταγωνιστών, πληρώνονται για την δουλειά αυτήν και έχουν στην διάθεση τους μεγάλη χρηματοδότηση κα πολύ προχωρημένα μέσα.
3. **Ιοί και κακόβουλο λογισμικό:** δεν είναι κάποιο φυσικό πρόσωπο που κάνει την επίθεση, είναι κάποιο πρόγραμμα το οποίο στοχεύει γενικά χωρίς να τον ενδιαφέρει ένας συγκεκριμένος χρήστης. Προσπαθούν να κάνουν ζημιές χωρίς να έχουν κάποιο συγκεκριμένο στόχο και μπορεί να προέρχονται από οπουδήποτε. Ένας χρήστης που δεν έχει εμπιστευτικές πληροφορίες στο σύστημα του θα πρέπει να φοβάται μόνο τους ιούς καθώς εξαιτίας τους μπορεί να καταρρεύσει όλο το

σύστημα, ενώ οι άλλοι εισβολείς δεν θα ενδιαφερθούν αφού δεν έχουμε σημαντικές πληροφορίες.

ΑΠΩΛΕΙΑ ΔΕΔΟΜΕΝΩΝ

Απώλεια δεομένων μπορεί να υπάρξει από εισβολείς, αλλά τις περισσότερες φορές συμβαίνει εξαιτίας κάποιου ατυχήματος, τέτοια ατυχήματα είναι:

- Θεομηνίες: πλημμύρες, σεισμοί, φωτιές
- Σφάλματα υλικού/λογισμικού: αλλοίωση
- Ανθρώπινα λάθη: διαγραφή αρχείων κατά λάθος

Τα προβλήματα αυτά αντιμετωπίζονται με αντίγραφα ασφαλείας τα οποία βρίσκονται κατά προτίμηση μακριά από το σύστημα. Τέτοιου είδους απώλεια δεδομένων είναι πολύ πιο σοβαρό πρόβλημα από τις παραβιάσεις.

ΑΡΧΕΣ ΚΡΥΠΤΟΓΡΑΦΙΑΣ

ΒΑΣΙΚΕΣ ΑΡΧΕΣ

Κρυπτογραφία: απόκρυψη δεδομένων

-Ξεκινάμε από ένα απλό κείμενο(P, plaintext)

-Εφαρμόζουμε έναν μετασχηματισμό

-Παράγουμε το κρυπτοκείμενο(C, ciphertext)

-Για τους τρίτους είναι ακατανόητο

Αλγόριθμοι κρυπτογράφησης: P->C

Αλγόριθμοι αποκρυπτογράφησης: C->P

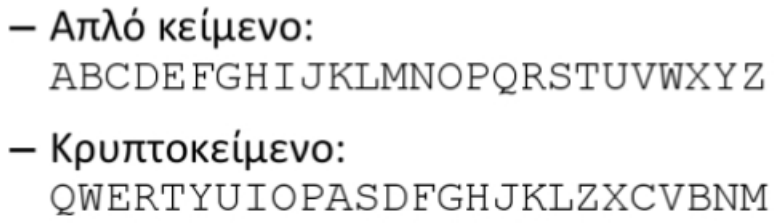
Αρχή του Kerchoff: οι αλγόριθμοι κρυπτογραφίας επειδή είναι δύσκολο να βρεθεί κάποιος καινούριος και μετά από κάποιο διάστημα σίγουρα θα μαθευτεί είναι δημόσιοι, αυτό που είναι μυστικό είναι τα κλειδιά τους τα οποία αλλάζουν και συχνά.

Ενναλακτικά: ασφάλεια μέσω κάλυψης. Αντίθετα με την αρχή του Kerchoff εδώ τα κρατάμε όλα μυστικά. Κάποια στιγμή όμως ο αλγόριθμος θα διαρρεύσει.

(ΑΛΓΟΡΙΘΜΟΙ)ΜΥΣΤΙΚΟΥ ΚΛΕΙΔΙΟΥ

Κρυπτογραφία μυστικού κλειδιού

-Μονοαλφαβιτική υποκατάσταση: κάθε γράμμα αντικαθιστάτε με ένα άλλο(1 προς 1).



Το κλειδί είναι η συμβολοσειρά αντικατάστασης. Η διαδικασία της κρυπτογράφησης είναι παρόμοια με της αποκρυπτογράφησης. Η αποκρυπτογράφησης όμως μπορεί να γίνει εύκολα αξιοποιώντας στατιστικά χαρακτήρων και οι επιθέσεις είναι απλές

-Κρυπτογραφία συμμετρικού κλειδιού: και τα δύο κλειδιά είναι ίδια ή το ένα παράγεται πολύ απλά από το άλλο, οπότε γνωρίζοντας μόνο το ένα κλειδί μπορούμε να βρούμε εύκολα το άλλο.

(ΑΛΓΟΡΙΘΜΟΙ)ΔΗΜΟΣΙΟΥ ΚΛΕΙΔΙΟΥ

Τα μυστικά κλειδιά έχουν πρόβλημα διανομής, δλδ δεν πρέπει να διαρρεύσουν. Πρέπει τα άτομα που επικοινωνούν να ανταλλάξουν τα κλειδιά και αυτό δεν μπορεί να γίνει με κρυπτογραφημένο τρόπο αφού στόχος μας είναι η κρυπτογράφηση, οπότε έχουμε θέμα.

Κρυπτογραφία δημόσιου κλειδιού.

Αντιμετωπίζουμε το πρόβλημα που είδαμε παραπάνω με ζεύγη κλειδιών. Έχουμε ένα **ιδιωτικό κλειδί** που διατηρείται μυστικό στον παραλήπτη και έχουμε το **δημόσιο κλειδί** που διανέμεται ανοιχτά σε όλους. Το δημόσιο χρησιμοποιείται για κρυπτογράφηση και το ιδιωτικό για αποκρυπτογράφηση. Τα κλειδιά δημιουργούνται και τα δύο μαζί αλλά η συσχέτιση τους είναι πάρα πολύ δύσκολη. Βασίζεται σε δύσκολα αντιστρέψιμες πράξεις. Οπότε έχοντας το κρυπτογραφημένο κείμενο κάποιου και το δημόσιο κλειδί δεν μπορείς να το αποκρυπτογραφήσεις.

ΜΟΝΟΔΡΟΜΕΣ ΣΥΝΑΡΤΗΣΕΙΣ

Μια συνάρτηση y=f(x) είναι μονόδρομη όταν:

- Ο υπολογισμός του y από το x είναι απλός
- Ο υπολογισμός του x από το y δεν είναι απλός, αλλά εξαιρετικά δύσκολος

Κρυπτογραφική συνάρτηση κατακερματισμού: μονόδρομη συνάρτηση με πρόσθετες ιδιότητες, αλλά δεν είναι συνάρτηση αποκρυπτογράφησης, οπότε οι μονόδρομες συναρτήσεις δεν μας κάνουν.

ΨΗΦΙΑΚΕΣ ΥΠΟΓΡΑΦΕΣ

Θέλουμε να συνδέσουμε ένα αρχείο με έναν οργανισμό, ένα άτομο, δλδ να ξέρουμε ότι προέρχεται από αυτόν. Αυτό που κάνει αυτός που κατέχει το αρχείο(δημιουργός) είναι να περνάει το αρχείο από μια μονόδρομη συνάρτηση και από αυτό παράγεται μια συμβολοσειρά σταθερού μήκους. Το αποτέλεσμα που πήραμε το κρυπτογραφούμε με το ιδιωτικό κλειδί το οποίο γνωρίζει μόνο ο δημιουργός. Το αποτέλεσμα των πράξεων αυτών είναι η ψηφιακή υπογραφή.

Στους τρίτους δίνεται το αρχικό κείμενο και η κρυπτογραφική συνάρτηση κατακερματισμού. Αυτοί περνάνε το κείμενο/αρχείο από την συνάρτηση και έπειτα αποκρυπτογραφούν και την υπογραφή, αν το αρχικό κείμενο μετά το πέρασμα από την συνάρτηση είναι ίδιο με αυτό που αποκρυπτογραφήσαμε τότε σημαίνει ότι η υπογραφή είναι έγκυρη.

Οι συναρτήσεις κρυπτογράφησης και της αποκρυπτογράφησης πρέπει να είναι συμμετρικές.

Μέχρι στιγμής υποθέσαμε ότι το δημόσιο κλειδί το γνωρίζουμε, αλλά δεν μπορούμε να γνωρίζουμε αν αυτό έχει πειραχθεί από κάποιον στην διαδρομή. Για να είμαστε βέβαιοι ότι το κλειδί προέρχεται από τον οργανισμό/άτομο που θέλαμε έχουμε ένα τρίτο έμπιστο άτομο που υπογράφει ψηφιακά το κλειδί. Πλέον υπάρχει η αρχή της πιστοποίησης, δλδ υπάρχει οργανισμός που υπογράφει δημόσια κλειδιά.

ΜΗΧΑΝΙΣΜΟΙ ΠΡΟΣΤΑΣΙΑΣ

ΤΟΜΕΙΣ ΠΡΟΣΤΑΣΙΑΣ

Για να καθορίσουμε ποιος μπορεί να κάνει τι στο σύστημα έχουμε τους τομείς προστασίας. Ο τομέας είναι ένα σύνολο ζεύγων από αντικείμενα και δικαιώματα. Δικαίωμα είναι η άδεια εκτέλεσης μιας λειτουργείας. Ο τομέας αναφέρεται σε έναν χρήστη ή μια ομάδα χρηστών. Στους τομείς πρέπει να υφίσταται η αρχή της ελάχιστης εξουσίας, δλδ ο τομέας

να διαθέτει τα ελάχιστα δυνατά δικαιώματα που του χρειάζονται για να κάνει ο χρήστης τις λειτουργίες που επιθυμεί. Μια διεργασία εκτελείται πάντα σε έναν τομέα.

Η διεργασία μπορεί να αλλάξει τομείς και αυτό μπορεί να γίνει με τους εξής τρόπους:

- Με κλίσεις συστήματος: πάμε σε επίπεδο πυρήνα
- Με την εκτέλεση προγράμματος SETUID/SETGID που αλλάζει τομέα.

Ο τομέας στον οποίο ανήκει μια διεργασία καθορίζεται από το UID και το GID.

Το ΛΣ επιβλέπει πολλά αντικείμενα. Κάθε αντικείμενο αντιπροσωπεύεται από ένα αρχείο.

Πως παρακολουθεί το σύστημα τα αντικείμενα και τους τομείς: έχουμε έναν πίνακα(μητρώο προστασίας) όπου οι γραμμές αντιστοιχούν σε τομείς και οι στήλες σε αντικείμενα, το κάθε κελί περιέχει τα δικαιώματα.

	Αντικείμενο							
	Αρχείο 1	Αρχείο 2	Αρχείο 3	Αρχείο 4	Αρχείο 5	Αρχείο 6	Εκτυπωτής 1	Σχεδιογράφος 2
Τομέας 1	Ανάγνωση	Ανάγνωση Εγγραφή						
2			Ανάγνωση	Ανάγνωση Εγγραφή Εκτέλεση	Ανάγνωση Εγγραφή		Εγγραφή	
3						Ανάγνωση Εγγραφή Εκτέλεση	Εγγραφή	Εγγραφή

Μπορούμε πάνω στον πίνακα αυτόν να μοντελοποιήσουμε τις εναλλαγές τομέων. Προσθέτουμε στις στήλες και τους τομείς και όπου υπάρχει η λέξη enter σημαίνει ότι μπορούμε από τον τομέα της γραμμής να πάμε στον τομέα της στήλης και να έχουμε τα ίδια δικαιώματα.

Επειδή ο μητρώο προστασίας είναι πολύ αραιό και μεγάλο στην πράξη δεν το αποθηκεύουμε έτσι αλλά το αποθηκεύουμε **ανά στήλες** οι γραμμές συμπιέζοντας το εκεί που δεν υπάρχουν κενά κελιά.

ΛΙΣΤΕΣ ΕΛΕΓΧΟΥ ΠΡΟΣΒΑΣΗΣ(ACL)

Λίστα δικαιωμάτων που αντιστοιχεί σε ένα αντικείμενο. Ουσιαστικά είναι μια στήλη του μητρώου προστασίας. Κάθε στοιχείο δίνει τομέα και δικαιώματα.

Οι τομείς αντιστοιχούν σε μεμονωμένους χρήστες(υποκείμενα/κύριοι) ή και σε ομάδες χρηστών. Ένας χρήστης μπορεί να ανήκει σε πολλές ομάδες χρηστών.

Υπάρχουν δικαιώματα ανά UID/GID combination. Ο συνδυασμός UID/GID λέγεται και ρόλος. Ένας χρήστης μπορεί να αποφασίσει να αλλάξει ρόλο δλδ ομάδα. Επίσης μπορεί ο

χρήστης να έχει και προνόμια ανεξαρτήτως ομάδας. Μπορούμε να έχουμε και επιλεκτικό αποκλεισμό ή αρνητικά προνόμια κάποιου τομέα από την πρόσβαση σε ένα αρχείο.

Τα προνόμια μπορεί να δίνονται σε επίπεδο χρήστη, σε επίπεδο ομάδας και για όλους μαζί.

Ο έλεγχος του ACL γίνεται κατά το «άνοιγμα» ενός αντικειμένου. Αν στην συνέχεια υπάρξει αποκλεισμός από το διάβασμα του αρχείου αυτό θα τεθεί σε ισχύ αφού «κλείσει» το αντικείμενο.

ΔΥΝΑΤΟΤΗΕΣ

Σε κάθε διεργασία/τομέα αντιστοιχεί λίστα δικαιωμάτων. Ουσιαστικά είναι μια γραμμή του μητρώου προστασίας και κάθε στοιχείο δείχνει αντικείμενο και δυνατότητες.

Οι λίστες ελέγχου πρόσβασης σχετίζονται με αντικείμενα οπότε είναι στον πυρήνα και δεν υπάρχει ο κίνδυνος να πειραχθούν τα δικαιώματα. Στην περίπτωση όμως που οι δυνατότητες σχετίζονται με την διεργασία υπάρχει ο κίνδυνος αυτή να πειράξει τα δικαιώματα. Πρέπει να βρούμε έναν τρόπο οι διεργασίες να έχουν πρόσβαση στις λίστες τους αλλά να μην μπορούν να τις τροποποιήσουν. Αυτό μπορεί να γίνει με τους εξής τρόπους:

- **Αρχιτεκτονικές με ετικέτες(tagged architectures):** σε κάθε λέξη αντιστοιχεί μια ετικέτα. Αν η τιμή της είναι 1 σημαίνει ότι η λέξη μπορεί να αλλάξει μόνο από το ΛΣ.
- **Διατήρηση των δυνατοτήτων μέσα στο λειτουργικό, δλδ σε επίπεδο πυρήνα.**
- **Κρυπτογράφηση των δυνατοτήτων:** όταν δημιουργείται το δικαίωμα για έναν χρήστη ο δημιουργός παράγει έναν τυχαίο πεδίο ελέγχου και το κρατάει τοπικά. Ο ιδιοκτήτης λαμβάνει ένα ειδικό αντικείμενο που έχει πληροφορίες για το ποιος δημιούργησε το δικαίωμα, το αντικείμενο και τις δυνατότητες. Επίσης με τις πληροφορίες αυτές με μια μονόδρομη συνάρτηση σε συνδιασμό με το πεδίο ελέγχου φτιάχνει μια κρυπτογραφική σύνοψη των στοιχείων. Τώρα όταν ο χρήστης στέλνει το αντικείμενο για να το χρησιμοποιήσει κρυπτογραφούνται ξανά οι πληροφορίες που θα στείλει και αν αυτές ταιριάζουν με την κρυπτογράφηση που έχει γίνει τότε του επιτρέπουμε να εκτελέσει την ενέργεια. Ο χρήστης μπορεί να αλλάξει τα δικαιώματα, αλλά μη γνωρίζοντας το πεδίο ελέγχου δεν μπορεί να πειράξει το αποτέλεσμα της μονόδρομης συνάρτησης και επομένως μπορούμε να κάνουμε τον έλεγχο για τον αν πείραξε τα δικαιώματα ή όχι.

Ανάκληση δικαιωμάτων χρήσης: όταν θέλουμε πολλοί τομείς να μην έχουν πλέον πρόσβαση σε ένα αντικείμενο θα πρέπει να πηγαίνουμε σε κάθε διεργασία και να αναιρέσουμε τα δικαιώματα που είχε, αυτό όμως είναι χρονοβόρο. Αυτό που κάνουμε είναι να δημιουργούμαι έμμεσα αντικείμενα τα οποία δείχνουν στα πραγματικά, αλλά οι διεργασίες αναφέρονται στα δικαιώματα που έχουν πάνω στα έμμεσα αντικείμενα. Οπότε

αν θέλουμε να κάνουμε ανάκληση δικαιωμάτων διαγράφουμε το έμμεσο αντικείμενο, δημιουργούμε ένα καινούριο και το κάνουμε έχουν πρόσβαση μόνο όσες διεργασίες θέλουμε.

Δυνατότητες ή λίστες ελέγχου πρόσβασης;

-Οι δυνατότητες επιτρέπουν γρήγορο έλεγχο.

-Οι λίστες ελέγχου πρόσβασης επιτρέπουν επιλεκτική ανάκληση.

Στα συστήματα γενικής χρήσης χρησιμοποιούνται λίστες ελέγχου πρόσβασης.

ΕΜΠΙΣΤΑ ΣΥΣΤΗΜΑΤΑ

Ένα σημαντικό ερώτημα είναι γιατί τα σύγχρονα ΛΣ έχουν προβλήματα ασφάλειας. Ενώ ξέρουμε πως να φτιάξουμε ένα ασφαλές ΛΣ υπάρχουν κάποιο παράγοντες που μας αποτρέπουν από το να το κάνουμε αυτό. Αρχικά για να φτιάξουμε ένα νέο ΛΣ που να είναι πλήρως προστατευμένο θα πρέπει να φτιάξουμε νέα προγράμματα και να ξεκινήσουμε από μια μηδενική βάση ενώ είναι πιο απλό απλά να αναβαθμίσουμε ένα ήδη υπάρχουν ΛΣ αφού θα χτίσουμε πάνω στα ήδη υπάρχοντα προγράμματα, άρα έχουμε το θέμα της συμβατότητας. Ένας άλλος σημαντικός παράγοντας είναι η πολυπλοκότητα των ΛΣ. Ένα ασφαλές ΛΣ θα είναι απλό, δεν θα δέχεται ενεργό περιεχόμενο από email(κάτι που μπορεί ο χρήστης να τρέχει) και δεν θα δέχεται ενεργό περιεχόμενο στις ιστοσελίδες. Αυτά είναι και οι τρόποι που μπορεί να παραβιαστεί η ασφάλεια του συστήματος μας.

Ο χρήστης έχοντας συνηθίσει να έχει στην διάθεση του αυτά τα περιεχόμενα ίσως δεν θα είναι πρόθυμος να αλλάξει σε έναν άλλον τρόπο λειτουργείας.

ΕΜΠΙΣΤΗ ΥΠΟΛΟΓΙΣΤΙΚΗ ΒΑΣΗ

Ορισμένοι οργανισμοί χρειάζονται ασφαλή συστήματα, όπως π.χ μια στρατιωτική βάση. Για να το πετύχουν αυτό τα συστήματα έχουν μια έμπιστη υπολογιστική βάση η οποία αποτελεί ένα μικρό τμήμα του λογισμικού και υλικού του συστήματος. Κάθε ενέργεια που μπορεί να προκαλέσει θέμα στην ασφάλεια του συστήματος περνάει πρώτα από αυτή την βάση και έπειτα να είναι ασφαλής θα συνεχιστεί. Θέματα μπορούν να προκαλέσουν συσκευές Ε/Ε, προγράμματα με προνόμια υπερχρήστη. Έχουμε τον ελεγκτή αναφορών που ελέγχει τις κλήσεις που γίνονται στο σύστημα.

ΤΥΠΙΚΑ ΜΟΝΤΕΛΑ(θεωρητικό μοντέλο ασφάλειας)

Μοντελοποιούμε την προστασία με 6 λειτουργίες:

-create object, delete object

-create domain, delete domain

- create right, delete right

Οι λειτουργίες συνδιάζονται σε διαταγές προστασίας:

-οι εφαρμογές εκτελούν διαταγές, όχι λειτουργίες

-παράδειγμα: δημιουργία νέου αρχείου

-συνδυάζει create object και create right

Ο το μητρώο προστασίας ξεκινάει από μια εξουσιοδοτημένη κατάσταση, αλλά αν το αλλάξουμε δεν γνωρίζουμε αν θα πάμε ξανά σε εξουσιοδοτημένη κατάσταση.

ΠΟΛΥΕΠΙΠΕΔΗ ΑΣΦΑΛΕΙΑ

Μοντέλα ασφάλειας

- **Επιλεκτικός έλεγχος πρόσβασης:** ο χρήστης αποφασίζει ποιος θα βλέπει τι. Συνηθισμένο στα συστήματα γενικής χρήσης.
- **Υποχρεωτικός έλεγχος πρόσβασης:** επιβολή κανόνων πρόσβασης από το σύστημα. Συνηθισμένο σε ευαίσθητα συστήματα όπως τα στρατιωτικά.

Μοντέλο Bell-La Padula

Σχεδιάστηκε για στρατιωτικά συστήματα. Έχουν διατεταγμένα επίπεδα διεργασιών αντικειμένων. Έχουμε δύο ιδιότητες ασφάλειας:

- **Απλή ιδιότητα ασφάλειας:** μια διεργασία στο επίπεδο κ μπορεί να διαβάσει <=κ
- **Ιδιότητα *:** μια διεργασία στο επίπεδο κ μπορεί να γράψει >=κ

Μοντέλο Biba

Το μοντέλο Bell-La Padula έχει σχεδιαστεί για να κρατάει μυστικά. Αναφορά προς τα πάνω και ενημέρωση από κάτω.

Το Biba εγγυάται την ακεραιότητα δεδομένων. Έχει δυο ιδιότητες:

- **Απλή ιδιότητα ακεραιότητας:** μια διεργασία στο επίπεδο κ μπορεί να γράψει <=κ
- **Ιδιότητα * ακεραιότητας:** μια διεργασία στο επίπεδο κ μπορεί να διαβάζει >=κ

Χρησιμοποιείται σε εταιρίες όπου ο διευθυντής δίνει εντολές στους εργαζόμενους.

ΣΥΓΚΕΚΑΛΥΜΜΕΝΑ ΚΑΝΑΛΙΑ

Το πρόβλημα του περιορισμού: πως μπορούμε να διασφαλίσουμε ότι η πληροφορία δεν φεύγει από εκεί που πρέπει να βρίσκεται με κανέναν τρόπο είτε εγκεκριμένο είτε μη εγκεκριμένο. Έστω ένας πελάτης ζητάει από τον διακομιστή κάποια εργασία. Για να γίνει η εργασία του αποκαλύπτει πληροφορίες. Ο διακομιστής θέλει να τις αποκαλύψει στον συνεργάτη αλλά το μητρώο προστασίας απαγορεύει την επικοινωνία. Υπάρχει ωστόσο τρόπος να επικοινωνήσει ο διακομιστής με τον συνεργάτη του χωρίς να περάσει από το σύστημα ασφάλειας του συστήματος και αυτός είναι με **συγκεκαλυμμένα κανάλια.** Τα κανάλια αυτά είναι γενικά θορυβώδη, δλδ είναι αδύνατον να πετύχουμε τον τέλειο συγχρονισμό μεταξύ διακομιστή και συνεργάτη. Μπορούμε να χρησιμοποιήσουμε κώδικα διόρθωσης σφαλμάτων για αυτό ή και χρήση επιβεβαιώσεων, δλδ να σταλθεί μήνυμα ότι λήφθηκε η πληροφορία και όλα καλά.

Κάτι άλλο που μπορούμε να κάνουμε είναι η **στεγανογραφία**, είναι μια μορφή συγκεκαλυμμένου καναλιού όπου κρύβουμε εμπιστευτικές πληροφορίες σε μη εμπιστευτικές με τροποποίηση πληροφοριών με μη εμφανή τρόπο.

ΠΙΣΤΟΠΟΙΗΣΗ ΤΑΥΤΟΤΗΤΑΣ

ΚΩΔΙΚΟΙ ΠΡΟΣΒΑΣΗΣ

Για να βεβαιωθούμε ότι ο χρήστης που λέει ότι είναι κάποιος είναι όντως αυτός χρειαζόμαστε μεθόδους πιστοποιήσεις. Υπάρχουν τρεις γενικοί μέθοδοι:

-κάτι που γνωρίζει ο χρήστη

-κάτι που έχει ο χρήστης

-κάτι που είναι ο χρήστης

Έχουμε τους **χάκερς** και **κράκερς:.**

Χάκερ-> είναι ένας επιδέξιος προγραμματιστής που μπορεί να λύσει δύσκολα προβλήματα.

Κράκερ-> είναι αυτός που διεισδύει τα συστήματα και σπάει την ασφάλεια τους.

Πιστοποίηση με χρήση κωδικών πρόσβασης

Χρειάζεται ένας συνδυασμός ονόματος χρήστη και κωδικού πρόσβασης. Οι κωδικοί πρόσβασης δεν πρέπει να εμφανίζονται για να μην τους δει κάποιος που βρίσκεται κοντά μας. Επίσης όταν κάνουμε LogIn δεν πρέπει να μας εμφανίζεται μνμ για το αν ήταν λάθος

το όνομα χρήστη ή ο κωδικός ώστε κάποιος που προσπαθεί να παραβιάσει το σύστημα να μην ξέρει που κάνει λάθος.

Πώς διεισδύουν οι κράκερς

Τα ονόματα χρήστη είναι συνήθως προφανή π.χ το ονοματεπώνυμο οπότε είναι πολύ απλό για τον κράκερ να έχει στη διάθεση του το όνομα χρήστη. Αλλά ακόμα και οι κωδικοί τις περισσότερες φορές είναι απλοί γιατί είναι κάτι που ο χρήστης θέλει να θυμάται, π.χ γενέθλια, όνομα παιδιών κτλ… Το πρόβλημα στην εύρεση/σπάσιμο κωδικών γίνεται ακόμα μεγαλύτερο εξαιτίας των δικτύων. Ο χρήστης στους πολλούς λογαριασμούς που έχει online χρησιμοποιεί συνήθως τον ίδιο κωδικό οπότε η εύρεση του ενός οδηγεί στην γνώση των υπολοίπων.

Στα πιο παλιά χρόνια ένας τρόπος για να αποκτούσε ο κράκερ πρόσβαση στο σύστημα είναι να καλεί συνέχεια τηλέφωνα μέσω ενός μόντεμ. Πολλοί υπολογιστές δεχόντουσαν τις κλήσεις χωρίς έλεγχο και έτσι ο κράκερ είχε πλέον πρόσβαση στο σύστημα. Η μέθοδος αυτή που χρησιμοποιούσαν οι κράκερς είναι γνωστή και ως **τηλεφωνητής πολέμου**.

Παραλλαγή της μεθόδου αυτής για το διαδίκτυο αποτελεί η τυχαία δοκιμή IP’s με ping και telenet και έτσι καταλαβαίνουν αν το IP είναι έγκυρο ή όχι. Το σύστημα μπορεί να πάρει κάποια αντίμετρα για αυτό. Μπορεί να απενεργοποιήσει το ping. Επίσης μετά από κάποιες αποτυχημένες προσπάθειες σύνδεσης ενός χρήστη να απενεργοποιήσει το telenet και να το ενεργοποιήσει ξανά αργότερα.

Ένας άλλος τρόπος είναι η σάρωση θυρών. Δοκιμάζει ο κράκερ να συνδεθεί με διάφορες θύρες γνωρίζοντας ήδη ότι κάποιος υπολογιστής βρίσκεται συνδεδεμένος σε μια θύρα. Στην συνέχεια βρίσκοντας τι είδους διεργασία εκτελείται κάνει συνέχεια επιθέσεις στις υπηρεσίες αυτές.

Άλλο τρόπο διείσδυσης αποτελεί η χρήση προεπιλεγμένων κωδικών. Στο σύστημα εξαρχής υπάρχουν λογαριασμοί ειδικού σκοπού των οποίων οι διαχειριστές δεν αλλάζουν τους κωδικούς , οπότε είναι σχετικά απλό για έναν τρίτο με γνώσεις να μπει στο σύστημα.

Άλλος τρόπος είναι το πρόγραμμα εξέτασης πακέτων(packet sniffer). Αυτό που γίνεται είναι να εγκαταστήσει ο κράκερ το πρόγραμμα αυτό στο σύστημα που έχει στοχεύσει και αυτό που κάνει το πρόγραμμα είναι να παρακολουθεί το δίκτυο και να καταγράφει τους κωδικούς και να τους στείλει πίσω στον κράκερ.

Άλλος τρόπος είναι τα σεναριόπαιδα(script kiddies). Αδαείς χρήστης βρίσκουν έτοιμα σενάρια για επίθεση και αξιοποιούν γνωστά προβλήματα και αδυναμίες του συτήματος.

Ασφάλεια κωδικών πρόσβασης στο UNIX

Η διατήρηση κωδικών σε αρχείο είναι επικίνδυνη. Γι'αυτό οι κωδικοί αποθηκεύονται κρυπτογραφημένοι με τη χρήση μια μονόδρομης συνάρτησης, οπότε όταν συνδεόμαστε κρυπτογραφείται ο κωδικός που δώσαμε και συγκρίνεται με τους κρυπτογραφημένους. Ο εισβολέας όμως μπορεί και αυτός να κρυπτογραφήσει τις λέξεις που θέλει να δοκιμάσει αν είναι κωδικοί και να τις συγκρίνει με τους κρυπτογραφημένους κωδικούς. Αυτό που κάνουμε για να αποφύγουμε το πρόβλημα αυτό είναι η χρήση της τεχνικής του αλατιού. Σε κάθε κωδικό αντιστοιχεί και ένας τυχαίος αριθμός ο οποίος κρυπτογραφείται μαζί με τον κωδικό. Τώρα ο κράκερ αφού προστέθηκε και ένας τυχαίος αριθμός θα χρειαστεί πολλές παραπάνω λέξει από αυτές που είχε για να βρει τον κωδικό. Επίσης η ανάγνωση των κωδικών γίνεται μέσω ενός ειδικού προγράμματος το οποίο καθυστερεί εσκεμένα να δώσει απάντηση. Ο χρόνος της καθυστέρησης είναι αμελητέος για τον χρήστη, αλλά επηρεάζει πολύ τον κράκερ.

Κωδικοί πρόσβασης μιας χρήσης

Μονόδρομη αλυσίδα κατακερματσιμού. Χρησιμοποιούμε μια μονόδρομη συνάρτηση y=f(x). Ο χρήστης επιλέγει τον μυστικό κωδικό(s) και το πλήθος των κωδικών που θέλει να παράξει(n). Αν θεωρήσουμε ότι το n=4 τότε ο πρώτος κωδικός θα είναι P1= f(f(f(f(s)))). Ο διακομιστής αυτό που κάνει είναι να ξεκινάει με P0=f(P1) όπου P0 η κρυπτογραφημένη μορφή του πρώτου κωδικού και 1 ο αριθμός που δίνει στον χρήστη για να ξέρει ότι ζητείται ο πρώτος κωδικός. Κατά την πρώτη σύνδεση ο διακομιστής στέλνει 1 και ο χρήστης P'1. Ο διακομιστής ελέγχει αν P0 = f(P'1) δλδ αν ο κωδικός που έδωσε ο χρήστης σε είναι σωστός. Αν είναι σωστός τότε αποθηκεύει τον P1 και την τιμή 2. Επειδή η συνάρτηση είναι μονόδρομη υπάρχει το καλό ότι ο διακομιστής δεν μπορεί να υπολογίσει τον προηγούμενο κωδικό. Ο χρήστης αφού έχει το s μπορεί να υπολογίσει όλους τους κωδικούς.

Πιστοποίηση ταυτότητας με ερωταποκρίσεις

Ο χρήστης δημιουργεί μια λίστα ερωτήσεων/απαντήσεων και οι απαντήσεις αποθηκεύονται σε κρυπτογραφημένη μορφή. Σε κάθε σύνδεση ο διακομιστής επιλέγει μια ερώτηση και ο χρήστης πρέπει να δώσει την σωστή απάντηση. Έτσι αποφεύγεται η συχνή έκθεση του κωδικού.

Παραλλαγή: ερωταπόκριση(challenge-response)

Ο χρήστης επιλέγει έναν αλγόριθμο f και έναν μυστικό κωδικό k. Σε κάθε σύνδεση ο διακομιστής στέλνει μια τιμή x και ο χρήστης απαντά με y=f(x,k). Ο εισβολές είναι σχεδόν σίγουρο ότι ξέρει τον αλγόριθμο f, αλλά είναι πολύ δύσκολο να ξέρει την τιμή κ που είναι μυστική οπότε δεν θα μπορεί να μαντέψει τον κωδικό μας και να συνδεθεί.

ΦΥΣΙΚΑ ΑΝΤΙΚΕΙΜΕΝΑ

Πιστοποίηση με χρήση φυσικού αντικειμένου

- Στο ATM έχουμε αντικείμενο(κάρτα) και PIN.
- Οι απλούστερες κάρτες έχουν μια μαγνητική λωρίδα, η κατασκευή όμως είναι πολύ οικονομική και αντιγράφονται εύκολα.
- Οι πιο σύνθετες κάρτες περιέχουν τσιπ μνήμης. Η αποθηκευμένη τιμή αλλάζει από τη συσκευή ανάγνωσης, π.χ τηλεκάρτες με αποθηκευμένο χρόνο ομιλίας.
- Ακόμη πιο σύνθετες είναι οι έξυπνες κάρτες. Το τσιπ περιέχει επεξεργαστή(μπορεί να εκτελεί αλγορίθμους) και μνήμη(μπορεί να αποθηκεύει και χρήματα). Η πιστοποίηση ταυτότητας με έξυπνες κάρτες είναι περίπου σαν του challenge-response.

ΒΙΟΜΕΤΡΙΑ

Αξιοποιεί μοναδικά χαρακτηριστικά του χρήστη. Συνήθως χρησιμοποιείται η ίριδα(διαφέρει ακόμα και στους δίδυμους) και τα μήκη δακτύλων χεριού. Αυτές οι τεχνικές βιομετρίας σπάνε εύκολα.

Άλλοι τρόποι πιστοποίησης στην βιομετρία είναι:

- Ανάλυση υπογραφής. Ο χρήστης γράφει το όνομα του στο τερματικό και αντί να γίνει έλεγχος του αποτελέσματος(πλαστογραφείται εύκολα) γίνεται έλεγχος των κινήσεων.
- Βιομετρία φωνής. Κάθε φορά που πάει ο χρήστης να συνδεθεί του δίνεται μια φράση να πει και το σύστημα αναλύει τα χαρακτηριστικά της φωνής του. Ζητώντας κάθε φορά διαφορετική πρόταση δεν μπορεί να χρησιμοποιηθεί μια έτοιμη ηχογράφηση.

ΕΣΩΤΕΡΙΚΕΣ ΕΠΙΘΕΣΕΙΣ

Επιθέσεις από χρήστες του συστήματος. Αφού ο χρήστης είναι μέσα στο σύστημα δεν χρειάζεται πιστοποιήσει ταυτότητας.

ΛΟΓΙΚΕΣ ΒΟΜΒΕΣ

Είναι κώδικας που ο προγραμματιστής των κρύβει. Όσο δουλεύει στην εταιρεία δεν γίνεται τίποτα αν απολυθεί όμως η βόμβα «εκρύγνυται» και μπορεί να αλλοιώσει αρχεία ή προγράμματα της εταιρείας. Έτσι ο προγραμματιστής μπορεί να εκβιάσει εύκολα την εταιρεία.

ΚΑΤΑΠΑΚΤΕΣ(trap doors)

Ο προγραμματιστής γράφοντας τον κώδικα μπορεί να παρακάμπτει κάποιους ελέγχους ασφαλείας, μπορεί να επιτρέπει σύνδεση στο σύστημα με κάποιο ειδικό password που έχει ορίσει ο προγραμματιστής. Η λύση στο πρόβλημα αυτό είναι να γίνονται code reviews, δλδ να ελέγχονται οι αλλαγές που οι προγραμματιστές έχουν κάνει στον κώδικα.

ΠΑΡΑΠΛΑΝΙΤΙΚΗ ΣΥΝΔΕΣΗ(login spoofing)

Χρησιμοποιείται σε δημόσιους υπολογιστές και υπάρχει ένα πρόγραμμα που παριστάνει την διαδικασία σύνδεσης. Ο χρήστης δίνει όνομα και κωδικό ανυποψίαστος και το πρόγραμμα αποθηκεύει τα στοιχεία του και του εμφανίζει μνμ λάθους και τερματίζει.

ΑΞΙΟΠΟΙΗΣΗ ΣΦΑΛΜΑΤΩΝ ΚΩΔΙΚΑ

ΣΦΑΛΜΑΤΑ ΚΩΔΙΚΑ

Κάθε σφάλμα είναι διαφορετικό, υπάρχουν όμως ορισμένα πολύ κοινά.

Ο επιτιθέμενος εκτελεί σάρωση θυρών για telenet. Στην συνέχεια δοκιμάζει να συνδεθεί μαντεύοντας όνομα και κωδικό. Αν τα καταφέρει τότε εκτελεί ένα πρόγραμμα. Δημιουργεί κέλυφος με προνόμια διαχειριστή. Στην συνέχει αυτό που κάνει είναι να κατεβάσει ένα πρόγραμμα ζόμπι το οποίο θα κάνει τις λειτουργίες που αυτός ζητάει. Η επίθεση θα μπορούσε να γίνει και με warms τα οποία πάνε από μηχανή σε μηχανή αν αποτύχουν.

ΥΠΕΡΧΕΙΛΙΣΗ ΜΝΗΜΗΣ

Υπερχείλιση προσωρινής μνήμης: η C δεν ελέγχει τα όρια των πινάκων κατά την πρόσβαση και έτσι ένα πρόγραμμα μπορεί να γράψει εκτός ορίων.

Πως λειτουργεί η επίθεση μέσω υπερχείλισης: έστω ότι το πρόγραμμα ζητάει κάποια παράμετρο σαν είσοδο και το πρόγραμμα έχει στην διάθεση του έναν πίνακα 1024 bytes. Αυτό που κάνει ο εισβολέας είναι να στείλει 2000 bytes. Αν δεν γίνει έλεγχος στα πόσα bytes στάλθηκαν τότε ο εισβολέας με τα δεδομένα που έστειλε αντικαθιστά τη διεύθυνση επιστροφής του προγράμματος. Το σημείο επιστροφής συνήθως είναι κάποιος κώδικας που έβαλε ο εισβολέας στην συμβολοσειρά εισόδου και έτσι ο εισβολέας παίρνει τον έλεγχο του προγράμματος.

Εντοπισμός των προβλημάτων υπερχείλισης: για να δούμε αν έχουμε κάποιο πρόβλημα υπερχείλισης δίνουμε στο πρόγραμμα μια μεγάλη είσοδο και αν καταρρεύσει εξετάζουμε την εικόνα μνήμης και βρίσκουμε το πρόβλημα στον πηγαίο κώδικα.

ΣΥΜΒΟΛΟΣΕΙΡΕΣ ΜΟΡΦΟΠΟΙΗΣΗΣ

Παρόμοια ιδέα με την υπερχείλιση. Στην C η εντολή printf() μπορεί να αλλάξει τα ορίσματα της. Έχουμε τον κωδικό %n που υπολογίζει πόσοι χαρακτήρες τυπώνονται, παράδειγμα: printf('hello %nworld\n', &i).

Ας πούμε ότι έχουμε το εξής σενάριο: το πρόγραμμα διαβάσει την συμβολοσειρά s και στην συνέχεια αντί να τυπώσει printf('%s',s) τυπώνει printf(s). Ο εισβολέας όμως μπορεί να έχει κωδικούς μορφοποίησης στην s και έτσι να καταφέρει να τροποποιήσει την μνήμη.

ΕΠΙΣΤΡΟΦΗ ΣΤΗΝ LIBC

Στις παραπάνω αξιοποιήσεις σφαλμάτων κώδικα που είδαμε στη στοίβα επιτρεπόταν να γίνει execute σε νέα όμως συστήματα δεν επιτρέπεται. Οπότε αυτό που γίνεται από τους εισβολείς είναι να γράφουν κανονικά τα δεδομένα τους με υπερχείλιση και στη συνέχεια να εκτελέσουν κώδικα της libc και συγκεκριμένα την strcpy() η οποία θα μεταφέρει το κομμάτι κώδικα που έγραψε ο χρήστης στην στοίβα σε κάποιο άλλο σημείο μνήμης που επιτρέπεται η εκτέλεση και θα αλλάξει και την διεύθυνση επιστροφής.

ΥΠΕΡΧΕΙΛΙΣΗ ΑΚΕΡΑΙΩΝ

Όλοι οι αριθμοί έχουν σταθερό μέγεθος. Στην Cδεν γίνεται έλεγχος για υπερχείλιση. Ο εισβολέας μπορεί να προκαλέσει υπερχείλιση. Έστω ότι ζητάει το πρόγραμμα ύψος και πλάτος. Θα γίνει κάποιος έλεγχος ώστε οι τιμές να είναι μεγαλύτερες από κάποιον αριθμό. Για να το αποφύγει αυτό ο εισβολέας προκαλεί υπερχείλιση ακεραίου και έτσι ξεφεύγει τον έλεγχο. Στη συνέχεια ο αριθμός που προέκυψε από την υπερχείλιση είναι πολύ μικρός και το πρόγραμμα δεσμεύει πολύ λίγη μνήμη και έτσι δίνεται η δυνατότητα επίθεσης υπερχείλισης προσωρινής μνήμης.

ΠΑΡΕΜΒΟΛΗ ΚΩΔΙΚΑ

Επίθεση με παρεμβολή κώδικα. Έστω ότι ένα πρόγραμμα χρησιμοποιεί την system(). Θέλουμε ο χρήστης να δώσει δύο ορίσματα, το αρχείο ανάγνωσης και το αρχείο εγγραφής. Αυτό που κάνει ο εισβολέας είναι να δώσει σαν είσοδο μια εκτελέσιμη εντολή η οποία θα προκαλέσει ζημιές στο σύστημα.

ΚΛΙΜΑΚΩΣΗ ΠΡΟΝΟΜΙΩΝ

Έχουμε τον cron που εκτελεί περιοδικά εργασίες για τους χρήστες. Εκτελείται με δικαιώματα διαχειριστή αλλά οι χρήστες δεν μπορούν να γράψουν σε αυτό. Ένας τρόπος να ξεγελάσει ο εισβολέας το ΛΣ είναι: εκτελεί το πρόγραμμα του στον κατάλογο του corn, αφήνει το πρόγραμμα του να καταρρεύσει και αυτό που κάνει το corn είναι να αποθηκεύσει

την εικόνα μνήμης του προγράμματος στον κατάλογο. Ο εισβολέας αφήνει μια εικόνα μνήμης που έχει την μορφή εντολών corn.

ΚΑΚΟΒΟΥΛΟ ΛΟΓΙΣΜΙΚΟ

ΕΙΔΗ ΚΑΚΟΒΟΥΛΟΥ ΛΟΓΙΣΜΙΚΟΥ(malware)

Κάποτε φτιάχνονταν από κακόβουλους ερασιτέχνες, αλλά πλέον φτιάχνεται από κακόβουλους επαγγελματίες.

Μπορεί να εγκαθιστά κερκόπορτα(πίσω πόρτα) στο σύστημα και να επιτρέψει σε τρίτους να μπούνε και να πάρουν τον έλεγχο της μηχανής. Η μηχανή μας πλέον γίνεται ζόμπι ή bot. Τα bot χρησιμοποιούνται για μαζικές εργασίες, π.χ μαζική αποστολή spam.

Μπορεί να εγκαθιστά καταγραφέα πληκτρολογήσεων που παρακολουθεί τι πληκτρολογούμε και οι κωδικοί που καταγράφει στέλνονται αλλού.

Μπορεί να κάνει υποκλοπή ταυτότητας(identity theft). Συλλέγει τα δεδομένα του χρήστη μέσω keylogger και στη συνέχεια μπορεί να παριστάνει ότι είναι ο χρήστης.

Γιατί υπάρχει τόσο κακόβουλο λογισμικό: Υπάρχουν λίγοι στόχοι, δλδ λειτουργικά. Και πολλά από τα λειτουργικά ή ακόμα και χρήστες θυσιάζουν την ασφάλεια για την ευχρηστία.

ΔΟΥΡΕΙΟΙ ΙΠΠΟΙ(trojan horses)

Συνήθως κακόβουλο λογισμικό κρυμμένο σε χρήσιμο πρόγραμμα το οποίο εγκαθιστά ο χρήστης π.χ πειρατικά προγράμματα.

Μια άλλη εναλλακτική για να τρέξουν αυτά τα προγράμματα είναι να τα εκτελέσει ο χρήστης κατά λάθος. Στα περισσότερα ΛΣ υπάρχει η μεταβλητή PATH και ο εισβολέας πάει και βάζει το πρόγραμμα του σε ένα εκτελέσιμο κατάλογο. Μπορεί ο χρήστης να πληκτρολογήσει λάθος εντολή και να τρέξει έτσι το κακόβουλο πρόγραμμα.

Κοινές απάτες με δούρειους ίππους:

-μεταφορά χρημάτων από λογαριασμό χρήστη.

-κλήση αριθμών πρόσθετης χρέωσης.

ΙΟΙ(viruses)

Είναι προγράμματα τα οποία αναπαράγονται μόνα τους. Προσκολλώνται σε εκτελέσιμα προγράμματα και το μολυσμένο πρόγραμμα προσπαθεί να μολύνει και άλλα.

Λειτουργεία ιών: Αρχικά ο δημιουργός μολύνει ένα πρόγραμμα, συνήθως επιλέγεται ένα δημοφιλές πρόγραμμα όπως π.χ τα πειρατικά τα οποία διαδίδονται και εύκολα. Κάθε φορά που εκτελείται το πρόγραμμα μολύνεται αυτό αφού έχει τον ιό και στην συνέχεια προσπαθεί να επεκταθεί και να μολύνει και άλλα προγράμματα.

Κατηγορίες ιών:

- **Συνοδευτικοί ιοί:** δεν προστίθενται στο πρόγραμμα αλλά εκτελούνται αντί αυτού και στη συνέχεια εκτελείται το πραγματικό πρόγραμμα. Για παράδειγμα στα windows μια συντόμευση μπορεί να δείχνει στον ιό και όποτε εκτελείται να εκτελείται πρώτα ο ιός και μετά το πραγματικό πρόγραμμα, δλδ η συντόμευση δείχνει στον ιό.
- **Ιοί εκτελέσιμου προγράμματος:** είναι η πιο συνηθισμένη κατηγορία ιών. Αλλάζουν το πρόγραμμα του εκτελέσιμου και τρέχουν μαζί με αυτό.
- **Ιοί αντικατάστασης:** αντικαθιστούν πλήρως το μολυσμένο πρόγραμμα και κρατάνε μόνο το δικό τους. Ωστόσο γίνονται γρήγορα αντιληπτοί αφού ο χρήστης παρατηρεί ότι το πρόγραμμα δεν κάνει τίποτα γι'αυτό προσπαθούν να μην μολύνουν αμέσως τα πάντα.
- **Παρασιτικοί ιοί:** προσαρτάται στο πρόγραμμα και το πρόγραμμα λειτουργεί κανονικά παρά τον ιό. Ο ιός μπαίνει στην αρχή, στο τέλος ή στη μέση του προγράμματος και συγκεκριμένα στα κενά που υπάρχουν για να μην αλλάζει το μέγεθος.
- **Ιοί τομέα εκκίνησης:** γράφεται στον τομέα εκκίνησης του λειτουργικού και εκτελείται κατά την εκκίνηση και εκείνη τη στιγμή έχει προνόμια υπερχρήστη. Ο ιός καταλαμβάνει κάποιο διάνυσμα το οποίο όμως εξαιτίας των διακοπών μπορεί και να το χάσει. Για να μην γίνει αυτό καταλαμβάνει πολλά διανύσματα και όταν εκτελείται ξανά καταλαμβάνει τα διανύσματα που έχασε.
- **Ιοί οδηγών συσκευών:** ο μολυσμένος οδηγός συσκευής είναι ο τέλειος ιός αφού εγκαθιστάτε από το ΛΣ και εκτελείται σε κατάσταση πυρήνα με ενισχυμένα προνόμια.
- **Μακροϊοί ή ιοί μακροεντολών:** μακροεντολές είναι μικρά κομμάτια εκτελέσιμου κώδικα. Ο ιός μπορεί για παράδειγμα να προσαρτηθεί στην εντολή OpenFile. Η διακίνηση του ιού είναι εύκολη, μπορούμε απλά να στείλουμε το word έγγραφο σε ένα άλλο χρήστη μέσω email.
- **Ιοί πηγαίου κώδικα:** μολύνουν τον πηγαίο κώδικα ενός προγράμματος.

Πώς εξαπλώνονται οι ιοί:

- Συχνά από μολυσμένο δωρεάν πρόγραμμα. Το πρόγραμμα διανέμεται μέσω ιστοσελίδων και όταν εκτελεστεί μπορεί να μολύνει ακόμα και τομείς εκκίνησης μηχανών.
- Μέσω email. Με ένα θέμα που τραβάει την προσοχή.

ΣΚΟΥΛΗΚΙΑ

Είναι παραλλαγή των ιών. Είναι κώδικας που κινείται αποκλειστικά μέσω δικτύου σε αντίθεση με των ιών που κινείται μέσω προγραμμάτων.

Το σκουλήκι του Internet(1988): βασίστηκε σε δύο κενά ασφάλειας στο BSD UNIX. Το σκουλήκι εκτελούσε τον βασικό κώδικα ο οποίος κατέβαζε τον κύριο κώδικα. Μετά δοκίμαζε να σπάσει όσους περισσότερους κωδικούς μπορούσε και μια φορά στις επτά έκανε και νέο αντίγραφο στη μηχανή.

ΛΟΓΙΣΜΙΚΟ ΚΑΤΑΣΚΟΠΙΑΣ

Φορτώνεται κρυφά από τον χρήστη στον υπολογιστή και προσπαθεί να μην δημιουργήσει πρόβλημα στο σύστημα. Κρύβεται ώστε να μην μπορεί να εντοπιστεί και συγκεντρώνει στοιχεία για τον χρήστη.

Μπορεί να έχει διάφορους στόχους:

-μάρκετινγκ: παρακολουθεί προτιμήσεις του χρήστη

-παρακολούθηση: παρακολουθεί ενέργειες του χρήστη

-Botnet: αφού παρακολουθήσει το σύστημα και κλέψει τους κωδικούς κάνει τη μηχανή ζόμπι.

Τρόποι εξάπλωσης:

-συνήθως κρύβεται σε χρήσιμα προγράμματα.

-μπορεί να κατεβαίνει αυτόματα από ιστοσελίδες.

-μερικές φορές ο χρήστης το αποδέχεται γιατί δεν διαβάσει τους όρους και συμφωνεί με αυτά.

Τι κάνει το λογισμικό κατασκοπείας:

-πειρατεία φυλλομετρητή: αλλάζει την σελίδα την αρχική σελίδα του φυλλομετρητή και τροποιεί τους σελιδοδείκτες του ώστε να κατευθύνει τον χρήστη προς συγκεκριμένες ιστοσελίδες.

-αλλαγή προεπιλεγμένης μηχανής αναζήτησης χρήστη.

-αντικατάσταση διαφημίσεων από ιστοσελίδες.

-εμφάνιση αναδυόμενων διαφημίσεων(pop-up)

-κλείσιμο τείχους προστασίας ή προστασίας από ιούς.

ROOTKIT

Πιο δύσκολη μορφή κακόβουλου λογισμικού για αφαίρεσης και προσπαθεί πολύ να κρυφτεί.

Έχουμε κάποιες κατηγορίες Rootkit:

- Rootkit υλικολογισμικού: κρύβεται στο BIOS

- Rootkit υπερεπόπτη: εκτελεί το κανονικό ΛΣ ως υπερεπόπτης

- Rootkit πυρήνα: κρύβεται σε οδηγό συσκευής ή στον πυρήνα

- Rootkit βιβλιοθήκης

- Rootkit εφαρμογής

Τα Rootkit υπερεπόπτη είναι δύσκολα στον εντοπισμό γιατί είναι αόρατα στο εκτελούμενο λειτουργικό σύστημα.

Τα Rootkit πυρήνα που είναι και τα πιο συχνά μπορούν να εντοπιστούν με την εκκίνηση του ΛΣ από USB και να γίνει σάρωση των αρχείων για αλλαγές.

Μπορούμε να απομακρύνουμε το Rootkit είτε αναιρώντας προσεκτικά τις αλλαγές είτε κάνοντας επανεγκατάσταση του συστήματος.

Ένα κλασικό παράδειγμα Rootkit αποτελεί αυτό της Sony BMG. Για να μπλοκάρει την παράνομη αντιγραφή των CD’s της είχε εγκατεστημένο ένα πρόγραμμα το οποίο εκτελούταν αυτόματα και είχε Rootkit. Το Rootkit μπλόκαρε τα προγράμματα ανάγνωσης και εγγραφής του PC και ο μόνος τρόπος να διαβάσουμε το CD ήταν μέσω ενός προγράμματος που έδινε η εταιρεία μαζί με το CD.Το Rootkit παρακολουθούσε τα πάντα και για λόγους παραβίασης ασφάλειας η εταιρεία καταδιώχθηκε νομικά.

ΤΡΟΠΟΙΟ ΑΜΥΝΑΣ

ΤΕΙΧΗ ΠΡΟΣΑΤΣΙΑΣ(firewall)

Προσφέρουν ασφάλεια σε βάθος: πολλά επίπεδα(αποτροπή εισόδου στο σύστημα, αποτροπή από το να λάβουν το σύστημα, αν το λάβουν τότε αντιμετώπιση).

Το διαδίκτυο εκθέτει το σύστημα σε κινδύνους και αυτό που κάνει το τείχος προστασίας είναι να αποτρέψει αυτούς τους κινδύνους ή έστω να τους μειώσει. Όλη η κίνηση από και προς το σύστημα ελέγχεται από το τείχος προστασίας και μόνο εγκεκριμένη κίνηση επιτρέπεται να περάσει.

Η υλοποίηση τους γίνεται με υλικό ή λογισμικό.

Τείχη προστασίας υλικού: όλη η κίνηση περνάει από το τείχος προστασίας. Το τι επιτρέπεται να περάσει ορίζεται με κανόνες.

Μη καταστατικό τείχος προστασίας: κάθε πακέτο ελέγχεται ανεξάρτητα από τα υπόλοιπα. Ελέγχεται η κεφαλίδα των πακέτων και οι κανόνες. Ισχύει πάντα ο πιο ειδικός κανόνας και αν δεν ισχύει κανένας τότε ισχύει ο τελευταίος ο οποίος απαγορεύει οτιδήποτε δεν επιτρέπεται. Το τείχος όμως δεν λύνει όλα τα προβλήματα διότι οι εξουσιοδοτημένες υπηρεσίες μπορεί να έχουν άλλα θέματα όπως σφάλμα υπερχείλισης. Όσο λιγότερες θύρες ανοίγουμε τόσο καλύτερα.

Καταστατικό τείχος προστασίας: παρακολουθεί την κατάσταση των συνδέσεων, δλδ κατά πόσο ένα πακέτο συνδέεται με σωστό τρόπο με τα προηγούμενα. Επιτρέπει συνδέσεις μόνο από μέσα προς τα έξω. Συνήθως έχει σύστημα ανίχνευσης εισβολής που ελέγχει το περιεχόμενο των πακέτων και ψάχνει για ύποπτα πακέτα.

Τείχη προστασίας λογισμικού: είναι προγράμματα τα οποία πρέπει εν μέρει να τρέχουν σε κατάσταση πυρήνα για να παγιδεύσουν όλη την κίνηση. Είναι κατάλληλα για προστασία ενός συστήματος.

ΠΡΟΣΑΤΣΙΑ ΑΠΟ ΙΟΥΣ(αντιμετώπιση)

Τεχνικές εναντίων των ιών: συνήθως μιλάμε για αντιβιοτικά(antivirus) τα οποία μπορούν να αντιμετωπίσουν και σκουλήκια και λογισμικό κατασκοπίας. Για κάθε τεχνική αντιβιοτικών που εφαρμόζει το σύστημα υπάρχει και ένα αντίμετρο από τους ιούς.

Σαρωτές ιών: οι κατασκευαστές μελετούν συνέχεια νέους ιούς. Καταγράφουν σε μια βάση τα χαρακτηριστικά των ήδη ιών και την ενημερώνουν τακτικά για νέους ιούς.

Ορισμένοι σαρωτές ελέγχουν τα μήκη των αρχείων και αν αυτά έχουν πειραχθεί τότε πολύ πιθανό στο σύστημα μας να υπάρχει ιός. Οι ιοί όμως γνωρίζοντας ότι εφαρμόζεται αυτή η τεχνική για την εύρεση τους αυτό που κάνουν είναι να εφαρμόσουν αντίμετρα.

Μπορούν να συμπιέσουν το πρόγραμμα για να μην αλλάξει το μέγεθος του, να κρυπτογραφήσουν τον κώδικα τους ώστε να μην μοιάζει με αυτών που υπάρχει στην βάση με τα χαρακτηριστικά ιών.

Πολυμορφικός ιός: αλλάζει λίγο σε κάθε αντιγραφή για να μην τον βρει ο σαρωτής. Χρησιμοποιεί μια μηχανή μετάλλαξης και αλλάζει τον κώδικα του. Μπορεί να προσθέσει κενές εντολές, εντολές που δεν έχουν αποτέλεσμα, περιττές εντολές ή να αλλάξει την σειρά τους.

Ο σαρωτής ελέγχει τον τομέα εκκίνησης και την μνήμη. Ο τομές εκκίνησης ελέγχεται ώστε να μην γίνει επανεκκίνηση του συστήματος έχοντας κάποιον ιό, ενώ η μνήμη για να ελέγχουμε να υπάρχει κάποιος ενεργός ιός. Αν όμως υπάρχει κάποιος ισχυρός ιός(όπως rootkit) τότε αυτός μπορεί να μας επιστρέψει λάθος στοιχεία για τα αρχεία και να νομίζουμε ότι είναι όλα καλά.

Ελεγκτές ακεραιότητας: υπολογίζουν αθροίσματα ελέγχου προγραμμάτων και σε κάθε εκτέλεση γίνεται σύγκριση με το αρχείο στο δίσκο. ο ελεγκτής μπορεί να υπογράψει ψηφιακά το αρχείο.

Ελεγκτές συμπεριφοράς: το αντιβιοτικό είναι συνεχώς στην μνήμη κα ελέγχει για ιούς. Κάνει έλεγχο σε όλες τις κλήσεις του συστήματος και παγιδεύει περίεργη συμπεριφορά. Ωστόσο δεν είναι προφανές τι είναι καλό και τι όχι.

ΑΠΟΦΥΓΗ ΙΩΝ

Οι χρήστες και τα οι εταιρείες μπορούν να κάνουν αρκετά για να αποφύγουν τους ιούς.

Αρχικά οι εταιρείες μπορούν να φτιάξουν απλά ΛΣ τα οποία είναι και πιο ασφαλή, πρέπει να προστατεύεται ο βασικός τομέας και η μνήμη flash.

Οι χρήστες δεν πρέπει να κατεβάζουν προγράμματα και λειτουργικά από μη αξιόπιστες πηγές καθώς αυτός είναι ένας από τους βασικούς τρόπους διάδοσης ιών. Πρέπει να ενημερώνουν το αντιβιοτικό, παίρνουν συχνά αντίγραφα ασφάλειας και όχι μόνο ένα καθώς μπορεί να είναι μολυσμένο.

ΥΠΟΓΡΑΦΗ ΚΩΔΙΚΑ

Πώς ξέρουμε ότι κάποιο λογισμικό είναι σωστό: υπογράφεται ψηφιακά από τον κατασκευαστή. Υπογράφεται η σύνοψη κώδικα με ιδιωτικό κλειδί και ο χρήστης επιβεβαιώνει την υπογραφή. Για να γίνει αυτό χρειάζεται μια μέθοδος διανομής δημόσιων κλειδιών.

ΦΥΛΑΚΙΣΗ

Αν δεν μπορούμε να βεβαιωθούμε ότι η πηγή που λάβαμε το λογισμικό είναι έμπιστη τότε αυτό το εκτελούμε σε φυλάκιση. Το πρόγραμμα είναι φυλακισμένο και εκτελείται υπό έλεγχο. Έχουμε τον **δεσμοφύλακα** που είναι ένα πρόγραμμα το οποίο ελέγχει τη κάνει το πρόγραμμα που κατεβάσαμε, ελέγχει τις κλήσεις του και επιτρέπει μόνο τις ασφαλείς. Μπορεί ο δεσμοφύλακας να υλοποιηθεί ως πρόγραμμα εκσφαλμάτωσης, δλδ σε debug mode.

ΑΝΙΧΝΕΥΣΗ ΕΙΣΒΟΛΗΣ

- Η ανίχνευση εισβολής μπορεί να γίνει με τεχνική φυλάκισης. Μοντελοποιούμε τα προγράμματα με γράφο κλήσεων που δείχνει ποια ακολουθία κλήσεων είναι επιτρεπτή. Ο δεσμοφύλακας παρακολουθεί τις κλήσεις και ένα δει ότι παρεκκλίνει από αυτές του γράφου τότε εντοπίζεται το πρόβλημα.
- **Δοχείο μελιού(honeypot):** έχουμε φτιάξει επίτηδες ένα σύστημα το οποίο είναι ελάχιστα προστατευμένο και «προσκαλούμε» το κακόβουλο λογισμικό κάνοντας το σύστημα αυτό να φαίνεται ενδιαφέρον και ότι περιέχει πολύ χρήσιμες και σημαντικές πληροφορίες, ενώ δεν είναι. Περιμένουμε στη συνέχεια να δεχτεί επιθέσεις. Έτσι βρίσκουμε με ποιο τρόπο ο εισβολέας βρήκε κενά στο σύστημα.

ΕΝΘΥΛΑΚΩΣΗ ΚΙΝΗΤΟΥ ΚΩΔΙΚΑ

Διάφορα είδη κινητού κώδικα(mobile code), κώδικας που κινείται από μηχανή σε μηχανή. Παραδείγματα αποτελούν μικροεφαρμογές σε ιστοσελίδες, πράκτορες που κινούνται στις μηχανές και αρχεία σε γλώσσα PostScript.

Το θέμα είναι να εκτελεστεί ο κώδικας με ασφάλεια. Τεχνικές για να γίνει αυτό είναι:

- **Αμμοπαγίδα(sandbox):** περιορίζει το πρόγραμμα στο να λειτουργεί σε ένα μικρό κομμάτι μνήμης και να μπορεί να χρησιμοποιεί μόνο συγκεκριμένες περιοχές διευθύνσεων. Το πρόγραμμα όμως κάνει και κλήσεις συστήματος, για να επιτραπούν οι κλήσεις περνάνε πρώτα από έναν ελεγκτή αναφορών και αν γίνει δεκτή τότε συνεχίζει αλλιώς μπορούμε και να το τερματίσουμε. Για να είμαστε σίγουροι ότι το πρόγραμμα θα τρέχει στην περιοχή μνήμης που έχουμε ορίσει για αυτό ελέγχουμε στατικά τις απόλυτες διευθύνσεις μνήμης. Για τις σχετικές διευθύνσεις μνήμης προσθέτουμε κώδικα πριν γίνει η αναφορά ώστε πρώτα να την ελέγξουμε.
- **Διερμηνεία(interpretation):** οι εντολές ελέγχονται κατά την διερμηνεία και αυτό είναι που κάνει η java. Ο διερμηνευτής λειτουργεί σαν αμμοπαγίδα. Όταν διερμηνεύουμε γίνεται αυτόματα έλεγχος από τον διερμηνευτή για τις αναφορές.

ΑΣΦΑΕΙΑ ΣΤΗ JAVA

Η JAVA ελέγχει αυστηρά τους τύπους και δεν επιτρέπει αυτόματη μετατροπή τους. Δεν επιτρέπει πρόσβαση σε δυναμικές δομές μέσω δεικτών και ελέγχει πλήρως την κατανομή της μνήμης.

Τα προγράμματα εκτελούνται σε JVM(Java Virtual Machine). Αρχικά μεταγλωττίζεται ο κώδικας σε ένα ενδιάμεσο στάδιο και από κει μπορεί είτε να διερμηνευτεί(δυναμικός έλεγχος) είτε να μεταγλωττιστεί(στατικός έλεγχος).

Οι εφαρμογές επαληθεύονται πριν εκετελεστούν. Έχουμε τον bytecode verifier που ελέγχει τον ενδιάμεσο κώδικα. Στατικός έλεγχος ώστε να δούμε ότι ο κώδικας δεν φτιάχνει δείκτες σε τυχαία σημεία της μνήμης, δεν πρέπει να βλέπει σε ιδιωτικά μέλη κλάσεων, να μην χρησιμοποιούνται τύποι λανθασμένα κτλ.

Οι κλήσεις συστήματος είναι ειδικές περιπτώσεις και ο τρόπος που θα τα διαχειριστούμε εξαρτάται από το επίπεδο JDK(Java Development Kit). Έχουμε τα εξής:

- **JDK 1.0:** οι έμπιστες εφαρμογές επιτρέπεται να κάνουν τα πάντα και έτσι είναι επικίνδυνες, ενώ οι μη έμπιστες(μικροεφαρμογές) δεν κάνουν τίποτα και έτσι είναι άχρηστες.
- **JDK 1.1:** έχουμε τις ψηφιακές υπογραφές. Αν μια εφαρμογή έχει υπογραφεί από έμπιστη οντότητα τότε είναι έμπιστη.
- **JDK 1.2:** είναι το πιο λεπτομερές μοντέλο ασφάλειας. Χρησιμοποιείται ένας πίνακας με προνόμια προσπέλασης. Σε κάθε γραμμή έχουμε την προέλευση της εφαρμογής και την οντότητα που την υπέγραψε, το αντικείμενο που μπορεί να εκτελέσει ενέργειες πάνω του και τις ενέργειες που μπορεί να κάνει.