1) Ποιές είναι οι βασικές λειτουργίες ενός λειτουργικού συστήματος;

1) Η διαχείριση των πόρων του υπολογιστή, όπως η CPU, η μνήμη, ο δίσκος, ο εκτυπωτής, κ.α.
2) Η δημιουργία διεπαφής χρήση
3) Η εκτέλεση και η παροχή υπηρεσιών για εφαρμογές λογισμικού

2) Ποιά τα είδη των πυρήνων;

1) Monolithic
Ένας μονολιθικός πυρήνας είναι μια αρχιτεκτονική λειτουργικού συστήματος όπου ολόκληρο το λειτουργικό σύστημα (το οποίο περιλαμβάνει τους drivers, το file system, την επικοινωνία διεργασιών) λειτουργεί στον χώρο του πυρήνα.
Είναι μεγάλο, πολύπλοκο, και όταν συμβαίνει κάποιο σφάλμα, τερματίζει όλο το λειτουργικό. Για μία μικρή αλλαγή σε ένα κομμάτι του, πρέπει να το κάνουμε compile ολόκληρο.

2) Microkernel
ένας μιρκοπυρήνας είναι το σχεδόν ελάχιστο ποσό λογισμικού που μπορεί να παρέχει τους μηχανισμούς που απαιτούνται για την υλοποίηση ενός λειτουργικού συστήματος. Αυτοί οι μηχανισμοί περιλαμβάνουν low-level address space management, thread management και inter-process communication (IPC).
Οι παραδοσιακές λειτουργίες του λειτουργικού συστήματος, όπως οι device drivers, protocol stacks και τα file systems, συνήθως απομακρύνονται από τον ίδιο τον πυρήνα και αντίθετα τρέχουν στο χώρο του χρήστη.

3) Layered Operating System
Ένα πολυεπίπεδο λειτουργικό σύστημα είναι ένα λειτουργικό σύστημα που συγκεντρώνει τη σχετική λειτουργικότητα μαζί και τη χωρίζει από τα μη σχετικά.

6) Operator (User)
5) User program
4) I/O Management
3) Device Driver
2) Memory Management
1) Process Allocation Management
0) Hardware

(4) Exokernel
Η αρχιτεκτονική exokernel έχει σχεδιαστεί για να διαχωρίζει την προστασία των πόρων από τη διαχείριση, για να διευκολύνει την προσαρμογή στις εφαρμογές.
Τα exokernels είναι συνήθως μικρού μεγέθους λόγω της περιορισμένης λειτουργικότητάς τους.
Εφαρμογές ή βιβλιοθήκες μπορούν να επικοινωνούν απευθείας με το hardware χωρίς να περάσουν από τον πυρήνα.

3) Περιγράψτε την ιεραρχία μνήμης.

Typical Access Time Typical Capacity

1 nsec	Registers	< 1 KB
2 nsec	Cache	~ 1+ MB
10 nsec	Main Memory	~ 1+ GB
10 msec	Magnetic Disk	~ 100+ GB
100 sec	Magnetic Tape	~ 100+ GB

4) Περιγράψτε την διαδικασία εκκίνησης.

1) Εκκινεί το BIOS και επιλέγει boot device, χρησιμοποιώντας πληροφορίες από την CMOS μνήμη
2) Εκτελεί τον κώδικα που βρίσκεται στο πρώτο sector (Master Boot Record) από το boot device
3) Διαπιστώνεται ποιο διαμέρισμα είναι ενεργό (active partition)
4) Διαβάζεται ο boot loader από το ενεργό διαμέρισμα
5) Εκκινεί το λειτουργικό σύστημα, το οποίο αρχικά φορτώνει τους οδηγούς συσκευών στον πυρήνα. Έπειτα δημιουργεί τις αρχικές διεργασίες και τέλος ξεκινάει την εκτέλεση ενός προγράμματος σύνδεσης χρήστη.

5) Τι είναι process, τι είναι thread και ποιά η διαφορά τους;

Process είναι το στιγμιότυπο ενός προγράμματος που εκτελείται σε έναν υπολογιστή. Μια διεργασία αποτελείται από το ίδιο το πρόγραμμα και από κάποιες τιμές που περιέχονται στη μνήμη και στους καταχωρητές του επεξεργαστή. Έχει ένα address space με stack, heap, text και global μεταβλητές.

Thread είναι η μικρότερη ακολουθία προγραμματισμένων εντολών που μπορεί να υποστεί ανεξάρτητη διαχείριση από έναν χρονο προγραμματιστή. Ένα νήμα είναι μια ελαφριά διεργασία.

Ένα νήμα διαφέρει από μια διεργασία ενός πολυεπεξεργαστικού λειτουργικού συστήματος στα εξής:

- οι διεργασίες είναι τυπικώς ανεξάρτητες, ενώ τα νήματα αποτελούν υποσύνολα μιας διεργασίας.

- οι διεργασίες περιέχουν σημαντικά περισσότερες πληροφορίες κατάστασης από τα νήματα, ενώ πολλαπλά νήματα μιας διεργασίας μοιράζονται την κατάσταση της διεργασίας όπως επίσης μνήμη και άλλους πόρους.

- οι διεργασίες έχουν ξεχωριστούς χώρους διευθυνσιοδότησης (address spaces), ενώ τα νήματα μοιράζονται το σύνολο του χώρου διευθύνσεων που τους παραχωρείται.

- η εναλλαγή ανάμεσα στα νήματα μιας διεργασίας είναι πολύ γρηγορότερη από την εναλλαγή ανάμεσα σε διαφορετικές διεργασίες.

6) Τι είναι το page table και τι το TLB;

Page table είναι η δομή δεδομένων που χρησιμοποιείται από ένα σύστημα εικονικής μνήμης σε ένα λειτουργικό σύστημα υπολογιστή για να αποθηκεύει το mapping μεταξύ εικονικών διευθύνσεων και φυσικών

διευθύνσεων.

TLB είναι μια προσωρινή μνήμη cache που χρησιμοποιείται για τη μείωση του χρόνου που απαιτείται για την πρόσβαση σε μια θέση μνήμης χρήστη. Είναι μέρος της μονάδας διαχείρισης μνήμης (MMU) του chip. Το TLB αποθηκεύει τις πρόσφατες μεταφράσεις της εικονικής μνήμης στη φυσική μνήμη.

7) Περιγράψτε το page fault και το TLB miss.

Page fault είναι ένας τύπος εξαίρεσης που προκαλείται από το hardware όταν ένα τρέχον πρόγραμμα ζητά πρόσβαση σε μία σελίδα μνήμης, για την οποία δεν υπάρχει mapping στον page table.
Η σελίδα ενδέχεται να είναι προσβάσιμη στο process, αλλά απαιτεί την προσθήκη mapping στο page table του process και μπορεί επιπλέον να απαιτεί το πραγματικό περιεχόμενο της σελίδας να φορτώνεται από το δίσκο.

TLB miss είναι η αποτυχία εύρεσης του mapping μιας εικονικής διεύθυνσης στο TLB.

8) Ποιά τα είδη των interrupts και ποιά η διαδικασία εκτέλεσης ενός interrupt;

1) Hardware interrupts
Εάν το σήμα για τον επεξεργαστή προέρχεται από εξωτερική συσκευή ή υλικό.

2) Software interrupts
Εάν το σήμα για τον επεξεργαστή προέρχεται από μία εντολή λογισμικού.

---

1) Το υλικό τοποθετεί στη στοίβα το PC, PSW, κτλ
2) Το υλικό φορτώνει το νέο PC από το διάνυσμα διακοπής
3) Η διαδικασία που είναι γραμμένη σε συμβολική γλώσσα αποθηκεύει τους registers
4) Η διαδικασία που είναι γραμμένη σε συμβολική γλώσσα δημιουργεί νέα στοίβα
5) Εκτελείται η εξυπηρέτηση διακοπής σε C
6) Ο scheduler αποφασίζει ποια διεργασία θα εκτελεστεί αμέσως μετά
7) Η Διαδικασία σε C επιστρέφει τον έλεγχο στην διαδικασία συμβολικής γλώσσας
8) Η διαδικασία συμβολικής γλώσσας ξεκινάει την εκτέλεση της νέας διεργασίας

9) Διαφορές μεταξύ kernel threads και user level threads.

User level threads
Ο πυρήνας δε γνωρίζει για την ύπαρξη τους.

Pros:
* Τα user level threads μπορούν να υλοποιηθούν ακόμα και αν ο πυρήνας δεν υποστηρίζει νήματα.

* Κάθε διεργασία έχει ένα πίνακα νημάτων για να παρακολουθεί τα νήματά της. Η εναλλαγή μεταξύ νημάτων είναι τουλάχιστον μία τάξη μεγέθους

ταχύτερη από τη μέθοδο που περιλαμβάνει παγίδευση στον πυρήνα.

* Επιτρέπουν σε κάθε διεργασία να διεθέτει το δικό της αλγόριθμο χρόνο προγραμματισμού. Μεγαλύτερη ευελιξία.

Cons:
* Εάν ένα user thread κάνει ένα system call, τότε θα μπλοκάρει και τα άλλα νήματα του ίδιου process.

* Εάν ένα thread προκαλέσει ένα page fault, τότε ο πυρήνας θα μπλοκάρει όλο το process, με αποτέλεσμα να μπλοκάρει και τα υπόλοιπα threads.

* Από τη στιγμή που ένα νήμα θα αρχίσει να εκτελείται, κανένα άλλο νήμα δεν μπορεί να εκτελεστεί μέχρι το εκτελούμενο να παραδώσει εκούσια τον έλεγχο της CPU.

* Ένα άλλο μειονέκτημα είναι πως οι προγραμματιστές χρειάζονται τα νήματα ακριβώς στις εφαρμογές όπου αυτά μπλοκάρονται πιο συχνά. Τα νήματα αυτά ενεργοποιούν συνεχώς κλήσεις συστήματος. Ο πυρήνας δεν βλέπει τα υπόλοιπα νήματα για να τα επιλέξει προς εκτέλεση και έτσι τα άλλα νήματα μένουν μπλοκαρισμένα χωρίς να παράγουν καθόλου έργο.

Kernel threads
Ο πυρήνας διατηρεί ένα πίνακα νημάτων ο οποίος παρακολουθεί όλα τα υπάρχοντα νήματα στο σύστημα.
Όταν ένα νήμα θέλει να δημιουργήσει ή να εξαλείψει ένα άλλο, πραγματοποιεί μια κλήση στον πυρήνα ο οποίος υλοποιεί τελικά τη δημιουργία ή την εξάλειψη ενημερώνοντας τα ταυτόχρονα τον πίνακα νημάτων.

Pros:
* Όταν μπλοκάρεται ένα νήμα πχ από ένα page fault, ο πυρήνας μπορεί να επιλέξει κάποιο άλλο προς εκτέλεση.

Cons:
* Το κόστος δημιουργίας, εξάλειψης και εναλλαγής είναι πολύ μεγάλο.
* Τι θα συμβεί κατά την εκτέλεση ενός fork(); Θα έχει η νέα διεργασία όλα τα νήματα της προηγούμενης ή όχι;
* Τι θα συμβεί εάν η διεργασία λάβει κάποιο signal; Ποιό νήμα θα το αναλάβει; Τι γίνεται αν 2 ή και παραπάνω δηλώσουν ενδιαφέρον;

10) Ποιά είναι τα states ενός process και ποιοί οι τύποι;

- States

1) Running
Τρέχει στη CPU
* Γίνεται Blocked πχ όταν περιμένει κάποιο I/O
* Γίνεται Ready όταν η CPU αρχίζει να εκτελεί μία άλλη διεργασία

2) Ready
Έτοιμη να τρέξη αλλά η CPU εκτελεί μία άλλη διεργασία
* Γίνεται Running όταν η CPU αρχίζει να εκτελεί αυτή τη διεργασία

3) Blocked
Περιμένει κάποιο εξωτερικό γεγονός
* Γίνεται Ready πχ όταν πραγματοποιηθεί το I/O

- Types

Χωρίζονται κυρίως σε:

- Foreground
Μια foreground process είναι οποιαδήποτε εντολή ή εργασία που εκτελείτε απευθείας και περιμένετε να ολοκληρωθεί. Ορισμένες foreground διεργασίες εμφανίζουν κάποιο είδος διεπαφής χρήστη που υποστηρίζει τη συνεχή αλληλεπίδραση του χρήστη

- Background
Μια background process είναι ένα πρόγραμμα που εκτελείται χωρίς είσοδο χρήστη.

Επιπρόσθετα, κατηγοριοποιούνται σε:

1) Zombie
Τερμάτισε τη λειτουργία αλλά υπάρχει ακόμα στο process table

2) Orphan
Η πατρική διεργασία τερμάτισε καθώς αυτή ακόμα εκτελείται

3) Daemon
Τρέχει χωρίς τον έλεγχο ενός χρήστη (non-interactive)

11) Ποιοί είναι οι 4 λόγοι τερματισμού ενός process;

1) Normal exit (voluntary) exit(0);
2) Error exit (voluntary) exit(1);
3) Fatal error (involuntary) exception
e.g. divide with zero
4) Killed by another process (involuntary)

12) Τι είναι η υβριδική υλοποίηση νημάτων και τι τα αναδυόμενα νήματα;

Είναι μία προσπάθεια να συνδυαστούν τα πλεονεκτήματα των kernel threads και των user level threads. Σε αυτή τη προσέγγιση ο προγραμματιστής μπορεί να επιλέξει πόσα kernel threads θέλει και με πόσα user threads θα πολυπλέξει καθένα από τα πρώτα. Ο πυρήνας γνωρίζει και χειρίζεται μόνο τα kernel threads, χωρίς να γνωρίζει για τα user threads. Αυτή η προσέγγιση προσφέρει μέγιστη ευελιξία.

Τα αναδυόμενα νήματα (pop-up threads) βρίσκουν χρήση στα κατανεμημένα συστήματα και αναλαμβάνουν το χειρισμό των εισερχόμενων μηνυμάτων. Ή έλευση ενός μηνύματος κάνει το σύστημα να δημιουργήσει ένα αναδυόμενο νήμα για τον χειρισμό του. Συνήθως χειρίζονται από τον πυρήνα, αν και έτσι μπορεί να προκαλέσει μεγάλη ζημιά.

13) Περιγράψτε τα race conditions.

Είναι η κατάσταση κατά την οποία 2 (ή και παραπάνω) threads/processes γράφουν την ίδια θέση μνήμης, την ίδια χρονική στιγμή (χωρίς συγχρονισμό).

14) Ποιές είναι οι συνθήκες για την ικανοποίηση του αμοιβαίου αποκλεισμού;

1) Το πολύ ένα process/thread στην κρίσιμη περιοχή
2) Δεν επιτρέπονται παραδοχές σε ό,τι αφορά την ταχύτητα ή το πλήθος των επεξεργαστών
3) Διεργασία που δε βρίσκεται σε κρίσιμο τμήμα δεν επιτρέπεται να μπλοκάρει άλλες διεργασίες
4) Δεν επιτρέπεται μια διεργασία να αναμένει επ' αόριστον να μπει στην κρίσιμη περιοχή της

15) Τι είναι οι κρίσιμες περιοχές;

Μία κρίσιμη περιοχή είναι το τμήμα ενός προγράμματος που έχει πρόσβαση σε κοινόχρηστους πόρους. Μόνο όταν μια διεργασία βρίσκεται στην κρίσιμη περιοχή της, μπορεί να είναι σε θέση να διακόψει άλλες διαδικασίες. Μπορούμε να αποφύγουμε τα race conditions εξασφαλίζοντας ότι δεν εισέρχονται δύο διεργασίες ταυτόχρονα στις κρίσιμες περιοχές.

16) Τι είναι τα semaphores και τι τα mutexes;

Semaphore είναι μία μεταβλητή που χρησιμοποιείται για τον έλεγχο πρόσβασης σε έναν κοινό πόρο από πολλαπλές διεργασίες σε ένα παράλληλο σύστημα
Τα semaphores που περιορίζονται στις τιμές 0 και 1 (ή κλειδωμένα / ξεκλείδωτα, μη διαθέσιμα / διαθέσιμα) ονομάζονται binary semaphores και χρησιμοποιούνται για την υλοποίηση locks.

Ένα mutex είναι ουσιαστικά το ίδιο με ένα binary semaphore και μερικές φορές έχει την ίδια υλοποίηση. Οι διαφορές μεταξύ τους είναι στον τρόπο με τον οποίο χρησιμοποιούνται.

17) Τι είναι ο scheduler;

Είναι το κομμάτι αυτό του λειτουργικού συστήματος το οποίο είναι υπεύθυνο για το ποια διεργασία εκτελείται ανά πάσα στιγμή. Μπορεί να υλοποιεί μία ή περισσότερες πολιτικές χρονο προγραμματισμού, ανάλογα με την κατηγορία των διεργασιών.

18) Ποιές είναι οι κατηγορίες αλγορίθμων χρόνο-προγραμματισμού;

1) Batch (Δέσμης)
Στα συστήματα δέσμης δεν υπάρχουν χρήστες που αδημονούν στα τερματικά τους για γρήγορες απαντήσεις. Διαθέτουν σε κάθε διεργασία μεγάλα διαστήματα. Η προσέγγιση αυτή μειώνει τις εναλλαγές ανάμεσα στις διεργασίες, άρα βελτιώνει την απόδοση.

Αλγόριθμοι:
* Εξυπηρέτηση με βάση τη σειρά άφιξης
* Εξυπηρέτηση με βάση τη μικρότερη διάρκεια
* Εξυπηρέτηση με βάση τη μικρότερη διάρκεια που απομένει

2) Interactive (Αλληλεπιδραστικά)
Συστήματα τα οποία χαρακτηρίζονται από σημαντικές αλληλεπιδράσεις μεταξύ του χρήστη και του υπολογιστή / συστήματος.

Αλγόριθμοι:
* Round robin
* Priority scheduling
* Πολλαπλές ουρές
* Εξυπηρέτηση με βάση τη μικρότερη διάρκεια
* Εγγυημένος προγραμματισμός
* Χρονοπρογραμματισμός με λοτταρία
* Χρονοπρογραμματισμός δίκαιης διανομής

3) Real-Time (Πραγματικού χρόνου)
Συστήματα που έχουν περιορισμούς πραγματικού χρόνου.

19) Ποιοί οι στόχοι ενός αλγορίθμου χρόνο-προγραμματισμού;

Για όλα τα συστήματα:

- Δικαιοσύνη
Να εκχωρείται σε κάθε διεργασία ένα δίκαιο μερίδιο της CPU

- Επιβολή της πολιτικής
Να παρακολουθείται αν εφαρμόζεται η καθορισμένη πολιτική

- Ισορροπία
Να διατηρούνται ενεργά όλα τα τμήματα του συστήματος

Επιπλέον για:

* Συστήματα δέσμης

- Διεκπεραιωτική ικανότητα
Να μεγιστοποιηθεί ο αριθμός των εργασιών που ολοκληρώνονται ανά ώρα

- Χρόνος διεκπεραίωσης
Να ελαχιστοποιηθεί ο χρόνος που μεσολαβεί ανάμεσα στην υποβολή και την ολοκλήρωση μιας εργασίας

- Αξιοποίηση της CPU
Να διατηρείται η CPU συνεχώς ενεργός

* Αλληλεπιδραστικά συστήματα

- Χρόνος απόκρισης
Η απόκριση στις αιτήσεις να είναι ταχύτατη

- Τήρηση αναλογιών
Να ικανοποιούνται οι προσδοκίες των χρηστών

* Συστήματα πραγματικού χρόνου

- Τήρηση των προθεσμιών
  Να αποφεύγεται η απώλεια δεδομένων

- Προβλεψιμότητα
  Να αποφεύγεται ο υποβιβασμός της ποιότητας στα συστήματα πολυμέσων

20) Ποιοί οι τρόποι επίτευξης mutual exclution;

1) Απενεργοποίηση διακοπών
   Θα απενεργοποιηθούν οι διακοπές ρολογιού καθώς και άλλες και έτσι η CPU δεν θα μπορεί να εναλλάξει την τρέχουσα διεργασία με κάποια άλλη.

2) Μεταβλητές κλειδώματος (locks)

3) Αυστηρή εναλλαγή

21) Τι είναι ο υποσιτισμός διεργασίας;

Είναι ένα πρόβλημα που συναντάται σε παράλληλα συστήματα όπου μια διαδικασία στερείται συνεχώς τους απαραίτητους πόρους για να επεξεργαστεί το έργο της

22) Γράψτε το αλγόριθμο του Peterson.

```
#define FALSE 0
#define TRUE  1
#define N     2

int turn;
int interested[N];

void enter_region(int process)
{
        int other;

        other = 1 - process;
        interested[process] = TRUE;
        turn = process;
        while (turn == process && interested[other] == TRUE);
}

void leave_region(int process)
{
        interested[process] = FALSE;
}
```

23) Busy-waiting (ενεργός αναμονή) σε μονο-επεξεργαστικά και σε πολυ-επεξεργαστικά συστήματα.

Είναι μια τεχνική που χρησιμοποιείται όταν προγραμματίζουμε με νήματα/threads. Η ιδέα είναι ότι το νήμα τρέχει ένα βρόχο επανάληψης μέχρι να ικανοποιηθεί κάποια συνθήκη για να συνεχίσει.

Συνήθως αποφεύγεται γιατί σπαταλάει χρόνο της CPU. Χρησιμοποιείται όταν υπάρχει σχετική βεβαιότητα ότι η αναμονή θα είναι σύντομη.

Σε μονο-επεξεργαστικά συστήματα, το busy-waiting αποφεύγεται λόγω του ότι εάν το κύριο process κάνει busy-wait τότε μπλοκάρει όλο το σύστημα.

Σε ένα πολυ-επεξεργαστικό σύστημα το busy-wait δεν θα παγώσει το σύστημα, ωστόσο και πάλι χάνονται κβάντα χωρίς παραγωγή έργου από το process.

24) Αναφέρετε χρήσεις του caching.

1) Μείωση του network traffic / wait-time
Κάνοντας cache δεδομένα δικτύου, δεν χρειάζεται να στέλνουμε ξανά πακέτα και να περιμένουμε απάντηση για δεδομένα που τα ζητήσαμε και στο σχετικά πρόσφατο παρελθόν (δεδομένου πως δεν έχουν υποστεί αλλαγές).

2) Μείωση των main memory lookups
Κάνοντας cache δεδομένα από προγράμματα, η CPU δεν χρειάζεται να τα ζητήσει από την κύρια μνήμη, με αποτέλεσμα τα προγράμματα να είναι πιο responsive.

3) Μείωση των page table lookups
Το TLB χρησιμοποιείται για να κάνει cache ορισμένα entries του page table.

25) Ποιό είναι το πρόβλημα αντιστροφής προτεραιοτήτων;

Είναι ένα σενάριο στον προγραμματισμό στον οποίο μια εργασία υψηλής προτεραιότητας μπλοκάρεται έμμεσα από μια εργασία χαμηλότερης προτεραιότητας που αντιστρέφει τις σχετικές προτεραιότητες
των δύο διεργασιών.
Αυτό παραβιάζει το μοντέλο προτεραιότητας, σύμφωνα με το οποίο οι εργασίες υψηλής προτεραιότητας μπορούν να μπλοκαριστούν μόνο από εργασίες υψηλότερης προτεραιότητας

26) Τι είναι το φράγμα;

Το φράγμα (barrier) είναι ένας μηχανισμός συγχρονισμού που προορίζεται κυρίως για ομάδες διεργασιών.
Όταν μία διεργασία φτάσει σε ένα φράγμα, περιμένει να φτάσουν στο ίδιο φράγμα και οι υπόλοιπες διεργασίες για να συνεχίσει.

27) Τι είναι η σελιδοποίηση και γιατί επινοήθηκε;

Είναι ένα σχήμα διαχείρισης μνήμης με το οποίο ένας υπολογιστής αποθηκεύει και ανακτά δεδομένα από δευτερεύουσα αποθήκευση, πχ δίσκο, για χρήση στην κύρια μνήμη. Σε αυτό το σχήμα, το λειτουργικό σύστημα ανακτά δεδομένα από το δίσκο σε μπλοκ ίδιου μεγέθους που ονομάζονται σελίδες.

Επινοήθηκε για να δημιουργηθεί μεγάλος και γραμμικός χώρος διευθύνσεων χωρίς να χρειάζεται να επεκταθεί η φυσική μνήμη

28) Περιγράψτε τη δομή μίας καταχώρισης στον πίνακα σελίδων.

A: Bit για την απενεργοποίηση της κρυφής μνήμης για τη σελίδα αυτή
B: Bit αναφοράς. Γίνεται set όταν το λειτουργικό αναφέρεται σε αυτή τη σελίδα, είτε για εγγραφή είτε για ανάγνωση
C: Bit τροποποίησης. Γίνεται set όταν το λειτουργικό γράφει σε αυτή τη σελίδα. Αναφέρετε και ως βρώμικο bit
D: Bit(s) προστασίας. Δηλώνουν ποια είδη πρόσβασης επιτρέπονται

* 1 bit:
- 0: Επιτρέπεται ανάγνωση/εγγραφή
- 1: Μόνο ανάγνωση
* 3 bit:
- 1ο bit: ανάγνωση
- 2ο bit: εγγραφή
- 3ο bit: εκτέλεση

E: Αριθμός πλαισίου σελίδας

29) Περιγράψτε τους αλγόριθμους αντικατάστασης σελίδων.

1) Βέλτιστος αλγόριθμος αντικατάστασης
Αδύνατον να υλοποιηθεί. Αντικαθιστά πάντα τη σελίδα που θα ζητηθεί στο πιο μακρινό μέλλον, με σκοπό να καθυστερήσει το περισσότερο δυνατό ένα σφάλμα σελίδας. Υποθετικός αλγόριθμος, χρησιμοποιείται ως μέτρο σύγκρισης.

2) Αλγόριθμος αντικατάστασης σελίδας NRU (Not Recently Used)
Αντικαθιστά σελίδες με βάση τις παρακάτω κατηγορίες:

0. Δεν έγινε αναφορά, δεν τροποποιήθηκε |
1. Δεν έγινε αναφορά, τροποποιήθηκε | προτεραιότητα
2. Έγινε αναφορά, δεν τροποποιήθηκε |
3. Έγινε αναφορά, τροποποιήθηκε ν

3) Αλγόριθμος αντικατάστασης FIFO

4) Αλγόριθμος αντικατάστασης σελίδας της δεύτερης ευκαιρίας
Παραλλαγή του FIFO οπού αντί να αντικαθίστανται απλά η παλαιότερη σελίδα, πρώτα εξετάζεται το bit αναφοράς. Εάν η σελίδα αναφέρθηκε πρόσφατα, ο αλγόριθμος επανατοποθετεί τη σελίδα στο τέλος της ουράς, δίνοντάς της έτσι μία δεύτερη ευκαιρία. Εάν όλες οι σελίδες αναφέρθηκαν πρόσφατα, ο αλγόριθμος μετατρέπεται σε έναν απλό FIFO αλγόριθμο.

5) Αλγόριθμος αντικατάστασης ρολογιού
Οι σελίδες οργανώνονται σε μία κυκλική λίστα. Υπάρχει ένα δείκτης που δεικτοδοτεί την παλαιότερη σελίδα. Σε ένα σφάλμα σελίδας έχουμε τα εξής:

- Bit αναφοράς 0: Αφαιρείται η σελίδα
- Bit αναφοράς 1: Ο δείκτης προχωρά μία θέση

6) Αλγόριθμος αντικατάστασης LRU (Least Recently Used)
Σε ένα σφάλμα σελίδας, αφαιρείται η σελίδα που δεν έχει χρησιμοποιηθεί για το μεγαλύτερο χρονικό διάστημα.
Αν και ο αλγόριθμος LRU είναι θεωρητικά υλοποιήσιμος, δεν είναι φθηνός.

7) Αλγόριθμος αντικατάστασης σελίδας του συνόλου εργασίας
Ο αλγόριθμος κάνει prepaging προσπαθώντας να διατηρεί στη μνήμη σελίδες που ανήκουν στο working set και να ελαχιστοποιήσει το πλήθος των page faults. Όπως και ο LRU, η υλοποίηση είναι κάπως ακριβή.

8) Αλγόριθμος αντικατάστασης σελίδας WSClock
Μοιάζει με τον αλγόριθμο του ρολογιού, όμως:

Σε ένα σφάλμα σελίδας ελέγχουμε τη σελίδα στην οποία δείχνει ο δείκτης:

- Αν η σελίδα έχει το bit αναφοράς 1, τότε το κάνουμε 0 και προχωράμε το δείκτη

- Αν η σελίδα έχει το bit αναφοράς 0:

  - Αν η σελίδα είναι καθαρή και ο χρόνος τελευταίας χρήσης είναι μεγαλύτερος από ένα τ, τότε η σελίδα δεν ανήκει στο working set, υπάρχει έγκυρο αντίγραφο στο δίσκο και έτσι η σελίδα αφαιρείται και αντικαθίστανται από τη νέα σελίδα.

  - Αν η σελίδα είναι βρώμικη, δεν μπορεί να αντικατασταθεί αμέσως, διότι δεν υπάρχει έγκυρο αντίγραφο στο δίσκο. Έτσι χρονο προγραμματίζεται η εγγραφή της στο δίσκο και ο δείκτης προχωρά με την επόμενη σελίδα.

Ο αλγόριθμος αυτός χρησιμοποιείται αρκετά και στην πράξη. Είναι αποδοτικός και εύκολος στην υλοποίηση.

30) Πώς χειρίζεται το σύστημα ένα σφάλμα σελίδας;

1) Παγίδευση στον πυρήνα και αποθήκευση του PC στη στοίβα
2) Μία ρουτίνα assembly αποθηκεύει τους registers, έπειτα καλεί το λειτουργικό σύστημα
3) Το λειτουργικό σύστημα εντοπίζει ποιά σελίδα χρειάζεται
4) Ελέγχει εάν η σελίδα είναι έγκυρη και επιτρέπεται η πρόσβαση σε αυτή
5) Αν η σελίδα είναι βρώμικη, χρονο προγραμματίζεται η εγγραφή της στο δίσκο, και γίνεται context switch για να τρέξει άλλη διεργασία μέχρι να ολοκληρωθεί η εγγραφή
6) Όταν η σελίδα είναι πλέον καθαρή, το λειτουργικό τη ζητάει από τον δίσκο και τη φέρνει στη μνήμη
7) Όταν η σελίδα φτάσει στη μνήμη, ενημερώνεται ο page table
8) Η διεργασία που προκάλεσε το σφάλμα χρονο προγραμματίζεται και το λειτουργικό επιστρέφει στην ρουτίνα συμβολικής γλώσσας
9) Η ρουτίνα αυτή ξανά φορτώνει του registers και επιστρέφει στο χώρο του χρήστη για να συνεχιστεί η εκτέλεση

31) Τι είναι η τμηματοποίηση και γιατί επινοήθηκε;

Είναι η τεχνική κατά την οποία ο χώρος εικονικών διευθύνσεων διαιρείται σε επιμέρους ανεξάρτητους χώρους διευθύνσεων, οι οποίοι λέγονται ονομάζονται (segments). Το κάθε τμήμα αποτελείται από μία γραμμική ακολουθία διευθύνσεων, από το 0 μέχρι κάποιο μέγιστο. Τα τμήματα μπορούν να αυξομειώνονται ανεξάρτητα χωρίς να επηρεάζονται.

Επινοήθηκε για να επιτρέπει στα προγράμματα και τα δεδομένα να χωρίζονται σε λογικά ανεξάρτητους χώρους διευθύνσεων, και να υποβοηθείται η κοινοχρησία και η προστασία.

32) Ποιά τα πλεονεκτήματα της τμηματοποίησης;

1) Απλούστευση του χειρισμού των δομών δεδομένων που αυξάνονται ή μειώνονται σε μέγεθος

2) Διευκολύνει την κοινή χρήση διαδικασιών ή δεδομένων από πολλές διεργασίες, πχ κοινόχρηστες βιβλιοθήκες (shared libraries)
Η βιβλιοθήκη μπορεί να τοποθετηθεί σε ένα τμήμα και να είναι διαθέσιμη σε πολλές διεργασίες, χωρίς να χρειάζεται να βρίσκεται στους χώρους διευθύνσεων όλων των διεργασιών.

33) Σε συστήματα που κατανέμουν τα αρχεία τμηματικά, ποιό το μειονέκτημα των μεγάλων blocks και ποιό αυτό των μικρών blocks;

- Μεγάλα blocks => σπατάλη χώρου (μικρά αρχεία σε μεγάλα blocks)
- Μικρά blocks => σπατάλη χρόνου (πολλές περιστροφές του δίσκου)

34) Ποιές οι διαφορές μεταξύ paging και segmentation.

1) Στο paging ο χρήστης δεν χρειάζεται να είναι ενήμερος πως χρησιμοποιείται η συγκεκριμένη τεχνική, στο segmentation χρειάζεται.

2) Στο paging υπάρχει 1 γραμμικός χώρος διευθύνσεων, στο segmentation πολλοί.

3) Στο paging οι διαδικασίες και τα δεδομένα δεν είναι διακριτά και δεν προστατεύονται ξεχωριστά, ενώ στο segmentation συμβαίνει.

4) Στο segmentation είναι ευκολότερος ο χειρισμός πινάκων μεταβλητού μεγέθους

5) Στο segmentation είναι ευκολότερη η κοινή χρήση των διαδικασιών ανάμεσα στους χρήστες

35) Ποιές είναι οι 3 θεμελιώδεις απαιτήσεις για μακρόχρονη αποθήκευση πληροφοριών;

1) Πρέπει να είναι δυνατόν να αποθηκεύονται μεγάλες ποσότητες πληροφοριών
2) Οι πληροφορίες πρέπει να επιβιώνουν μετά από τον τερματισμό της διεργασίας που τις χρησιμοποιεί

3) Πρέπει να επιτρέπεται σε πολλές διεργασίες να έχουν τη δυνατότητα να προσπελάσουν ταυτόχρονα τις πληροφορίες

36) Τι είναι τα αρχεία και ποιοί οι τύποι τους;

Είναι λογικές μονάδες πληροφοριών τις οποίες δημιουργούν οι διεργασίες

Τύποι:

1) Κανονικά αρχεία
Περιέχουν πληροφορίες χρήστη

2) Κατάλογοι
Βοηθούν στη δόμηση του συστήματος αρχείων

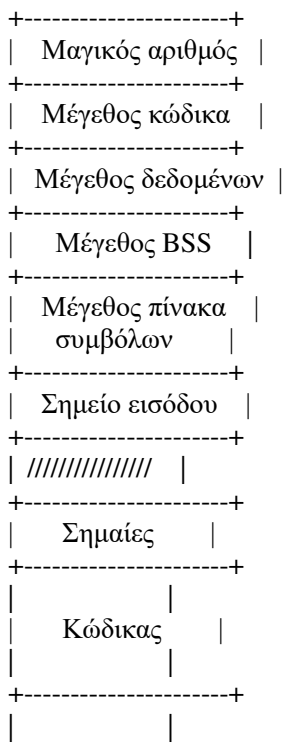
3) Αρχεία χαρακτήρων
Σχετίζονται με την είσοδο / έξοδο και χρησιμοποιούνται για να μοντελοποιήσουν σειριακές συσκευές Ε/Ε όπως τα τερματικά, εκτυπωτές, και τα δίκτυα.

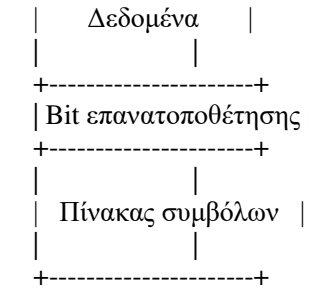
4) Αρχεία μπλοκ
Χρησιμοποιούνται για να μοντελοποιήσουν δίσκους

37) Τι είναι ο μαγικός αριθμός (magic number);

Είναι ο αριθμός στην κεφαλίδα ενός αρχείου που υποδηλώνει πως το αρχείο αυτό είναι εκτελέσιμο.

38) Ποιά η μορφή ενός εκτελέσιμου αρχείου UNIX;





39) Αναφέρετε μερικά χαρακτηριστικά αρχείων.

1) Current size
2) Time of last access
3) Time of last change
4) Creator
5) Owner
6) Protection / permissions
7) Hidden flag
8) Creation time
9) Password
...

40) Αναφέρετε μερικές λειτουργίες αρχείων.

1) Create
2) Delete
3) Open
4) Close
5) Read
6) Write
7) Append
8) Seek
9) Rename
...

41) Περιγράψτε τις μεθόδους αποθήκευσης αρχείων.

1) Συνεχής κατανομή
Η πιο απλή μέθοδος κατανομής. Κάθε αρχείο αποθηκεύεται ως μία συνεχόμενη αλληλουχία μπλοκ δίσκου.

Pros:

1) Απλή στην υλοποίηση. Για την εύρεση των μπλοκ ενός αρχείου στο δίσκο χρειαζόμαστε μονάχα τη διεύθυνση του 1ου μπλοκ και τον αριθμό των μπλοκ του αρχείου

2) Η απόδοση κατά την ανάγνωση είναι εξαιρετική, επειδή μπορεί να διαβαστεί ολόκληρο το αρχείο από το δίσκο με μία λειτουργία

Cons:

1) Με την πάροδο του χρόνου, ο δίσκος κατατέμνεται. Έτσι ο δίσκος θα καταλήξει να περιέχει αρχεία και κενά τμήματα. Για να γεμίσει τα κενά αυτά, θα πρέπει ο προγραμματιστής να δηλώνει πριν επεξεργαστεί ένα αρχείο, ποιό θα είναι το μέγεθος του τελικού αρχείου, προκειμένου να μπορέσει το λειτουργικό να βρει μία θέση για να το αποθηκεύσει. Αυτό καθιστά την συνεχή κατανομή μη πρακτική.

Η συνεχής κατανομή βρίσκει χρήσεις στην αποθήκευση αρχείων σε CD-ROM, όπου το μέγεθος των δεδομένων είναι γνωστό εκ των προτέρων.

2) Κατανομή συνδεδεμένης λίστας
Τα μπλοκ του κάθε αρχείου βρίσκονται σε μία συνδεδεμένη λίστα. Το κάθε μπλοκ περιέχει ένα δείκτη στη διεύθυνση του επόμενου κατά σειρά μπλοκ. Ο κατάλογος μπορεί να αποθηκεύσει το δείκτη του 1ου μπλοκ ως αφετηρία.

Pros:

1) Δεν υπάρχει απώλεια χώρου εξαιτίας κατάτμησης

Cons:

2) Η τυχαία προσπέλαση είναι πάρα πολύ αργή. Για να διαβαστεί ένα αρχείο θα πρέπει να γίνουν τόσες τυχαίες προσπελάσεις όσα και τα μπλοκ του αρχείου. Επομένως οι αναγνώσεις είναι εξαιρετικά χρονοβόρες.

3) Κατανομή συνδεδεμένης λίστας με χρήση πίνακα στη μνήμη
Εδώ επεκτείνουμε την κατανομή συνδεδεμένης λίστας και χρησιμοποιούμε ένα Πίνακα Κατανομής Αρχείων (FAT - File Allocation Table). Ο πίνακας αυτός αποθηκεύει τους δείκτες όλων των μπλοκ των αρχείων στη μνήμη.

Pros:

Η προσπέλαση της αλυσίδας, δηλαδή των δεικτών των μπλοκ, γίνεται χωρίς τυχαία προσπέλαση στο δίσκο. Εάν για παράδειγμα θέλουμε το 100ο μπλοκ ενός αρχείου, η διεύθυνση αυτού θα βρίσκεται στη μνήμη, επομένως δε χρειάζεται να εκτελέσουμε 100 τυχαίες προσπελάσεις στο δίσκο για να τη βρούμε. Για την προσπέλαση του ίδιου του μπλοκ ωστόσο, θα πρέπει να μπούμε στο δίσκο.

Cons:

Η αποθήκευση ολόκληρου του πίνακα στη μνήμη.

4) Κόμβοι i
Μία μέθοδος που συσχετίζει κάθε αρχείο με ένα κόμβο i, μία δομή δεδομένων που περιέχει τα χαρακτηριστικά και τις διευθύνσεις των μπλοκ του αρχείου στο δίσκο (όχι όλες). Ο κόμβος i πρέπει να βρίσκεται στη μνήμη μονάχα όταν το αρχείο είναι ανοιχτό.

Pros:

1) Ο κόμβος i είναι συνήθως μικρότερος από το FAT.

2) Αρχεία που δεν είναι ανοιχτά, δεν καταλαμβάνουν χώρο στη κύρια μνήμη

Cons:

1) Επιβάλλεται μία ιεραρχία από μπλοκ διευθύνσεων εάν τα αρχεία αποτελούνται από πολλά μπλοκ. Το τελευταίο κελί του κόμβου i θα περιέχει τη διεύθυνση ενός τέτοιου μπλοκ διευθύνσεων.

42) Περιγράψτε τα συστήματα αρχείων με καταγραφική δομή (LFS).

Όλες οι εγγραφές τοποθετούνται σε προσωρινή μνήμη, και γράφονται περιοδικά στο δίσκο σε ένα μόνο τμήμα, στο τέλος του αρχείου καταγραφής (log). Κατά το άνοιγμα ενός αρχείου χρησιμοποιείται ο χάρτης για να εντοπιστεί ο κόμβος i για το αρχείο. Από τη στιγμή που θα εντοπιστεί ο κόμβος i, οι διευθύνσεις του μπλοκ μπορούν να βρεθούν από αυτόν. Όλα τα μπλοκ βρίσκονται σε τμήματα κάπου μέσα στο αρχείο καταγραφής.
Το LFS χρησιμοποιεί ένα νήμα καθαρισμού (cleaner thread) που σαρώνει συνεχώς κυκλικά το αρχείο καταγραφής για να το συμπτύξει.

43) Περιγράψτε τα εικονικά συστήματα αρχείων (VFS)

Η βασική ιδέα είναι να δημιουργηθεί μια αφαίρεση εκείνου του μέρους του συστήματος αρχείων που είναι κοινό σε όλα τα συστήματα αρχείων και να τοποθετηθεί ο κώδικάς τους σε ένα ξεχωριστό επίπεδο το οποίο καλεί τα υποκείμενα πραγματικά συστήματα αρχείων για να χειριστούν τα δεδομένα.

Στόχος είναι να ενσωματώσουν πολλά συστήματα αρχείων σε μία τακτοποιημένη δομή.

Το VFS διαθέτει δύο ξεχωριστές διασυνδέσεις:

1) Την ανώτερη με τις διεργασίες χρήστη
2) Την κατώτερη με τα πραγματικά συστήματα αρχείων

44) Περιγράψτε την αναζήτηση του ονόματος διαδρομής /usr/ast/mbox

1) Ανάγνωση του κόμβου i για το βασικό κατάλογο (κόμβος i 1)
2) Ανάγνωση του βασικού καταλόγου (μπλοκ 1)
3) Ανάγνωση του κόμβου i για τον κατάλογο /usr
4) Ανάγνωση του καταλόγου /usr
5) Ανάγνωση του κόμβου i για τον κατάλογο /usr/ast
6) Ανάγνωση του καταλόγου /usr/ast
7) Ανάγνωση του κόμβου i /usr/ast/mbox
8) Ανάγνωση του αρχείου /usr/ast/mbox

Για κάθε κατάλογο/αρχείο, χρειάζομαι 1 read για το i node και 1 read για τα blocks του καταλόγου/αρχείου.

45) Τι είναι ο ελεγκτής συσκευών (device controller);

Αποτελεί το ηλεκτρονικό κομμάτι μίας μονάδας Ε/Ε.
Μεταφράζει τα σήματα που πηγαίνουν και προέρχονται από την CPU.

46) Τι είναι η Ε/Ε με χαρτογράφηση στη μνήμη (Memory Mapped I/O);

Αποτελεί μία μέθοδο για εκτέλεση εισόδου/εξόδου (I/O) μεταξύ της κεντρικής μονάδας επεξεργασίας (CPU) και των περιφερειακών συσκευών σε έναν υπολογιστή.
Χρησιμοποιεί τον ίδιο χώρο διευθύνσεων για την αντιμετώπιση τόσο των συσκευών μνήμης όσο και των συσκευών I/O. Έτσι, όταν η CPU αποκτά πρόσβαση σε μια διεύθυνση, μπορεί να αναφέρεται σε ένα τμήμα της φυσικής μνήμης RAM, ή μπορεί να αναφέρεται στη μνήμη της συσκευής εισόδου / εξόδου.

47) Τι είναι η άμεση προσπέλαση μνήμης (DMA);

Είναι μια λειτουργία των συστημάτων υπολογιστών που επιτρέπει σε ορισμένα υποσυστήματα υλικού να έχουν πρόσβαση στην κύρια μνήμη του συστήματος (μνήμη τυχαίας προσπέλασης), ανεξάρτητα από την κεντρική μονάδα επεξεργασίας (CPU).
Είναι απαραίτητη η παρουσία ενός ελεγκτή DMA.

48) Σχολιάστε για ακριβής vs ανακριβής διακοπές.

Ακριβής διακοπές:
Μια διακοπή που αφήνει τη μηχανή σε καλά ορισμένη κατάσταση ονομάζεται ακριβής διακοπή (precise interrupt) και έχει 4 ιδιότητες:

1) Ο PC αποθηκεύεται σε γνωστή θέση
2) Όλες οι εντολές πριν από αυτή στην οποία δείχνει ο PC έχουν ολοκληρωθεί
3) Δεν έχει εκτελεστεί καμία εντολή μετά από αυτή στην οποία δείχνει ο PC
4) Η κατάσταση εκτέλεσης της εντολής στην οποία δείχνει ο PC είναι γνωστή

Ανακριβής διακοπές:
Οι διακοπές που δεν πληρούν τις παραπάνω προδιαγραφές ονομάζονται ανακριβείς διακοπές.
Όταν συμβαίνει μία ανακριβής διακοπή, το σύστημα συνήθως αδειάζει μεγάλες ποσότητες πληροφοριών της εσωτερικής του κατάστασης στη στοίβα, ώστε να δώσει στο λειτουργικό σύστημα τη δυνατότητα να εξακριβώσει τι ακριβώς συμβαίνει.

49) Ποιοί είναι οι στόχοι του λογισμικού Ε/Ε;

1) Ανεξαρτησία από τη συσκευή (device independence)
Να είναι δυνατή η γραφή προγραμμάτων τα οποία θα μπορούν να προσπελάσουν οποιαδήποτε συσκευή Ε/Ε χωρίς να χρειάζεται να καθορίζεται εκ των προτέρων η συσκευή.

2) Ομοιόμορφη ονομασία (uniform naming)
Το όνομα ενός αρχείου ή μίας συσκευής πρέπει να είναι απλώς μία συμβολοσειρά ή ένας ακέραιος και να μην εξαρτάται καθόλου από την εκάστοτε συσκευή.

3) Χειρισμός σφαλμάτων (error handling)
Να συμβαίνει όσο το δυνατόν πλησιέστερα στο υλικό.

4) Προσωρινή αποθήκευση (buffering)

50) Ποιοί είναι οι τρόποι πραγματοποίησης Ε/Ε;

1) Προγραμματισμένη Ε/Ε

Συχνά υλοποιείται με:
- Περιόδευση (polling) ή
- Αναμονή με απασχόληση (busy waiting)

Pros:

1) Είναι απλή

Cons:

1) Δεσμεύει τη CPU

2) Ε/Ε οδηγούμενη από διακοπές

Αντί η CPU να περιμένει τη μονάδα Ε/Ε να τελειώσει, αφού της στείλει τα δεδομένα, καλεί τον χρονο προγραμματιστή και αρχίζει να εκτελεί μία άλλη διεργασία. Μόλις η μονάδα Ε/Ε τελειώσει τη δουλειά της και είναι έτοιμη πάλι να δεχτεί δεδομένα από τη CPU, παράγει ένα interrupt το οποίο θα διακόψει τη CPU. Αυτή στέλνει εκ νέου δεδομένα και ξανά καλεί το χρονο προγραμματιστή για να εκτελέσει κάποια άλλη διεργασία, κοκ.

Pros:

1) Καλύτερη διαχείριση του χρόνου της CPU

Cons:

1) Οι διακοπές είναι γενικά χρονοβόρες

3) Ε/Ε με τη χρήση DMA

Η ιδέα εδώ είναι να ανατεθεί στον ελεγκτή DMA η τροφοδοσία της μονάδας Ε/Ε, χωρίς να απασχολείται η CPU. Στην ουσία, η DMA είναι προγραμματισμένη Ε/Ε, μόνο που όλη η δουλειά γίνεται από τον ελεγκτή DMA αντί από τη CPU.

Pros:

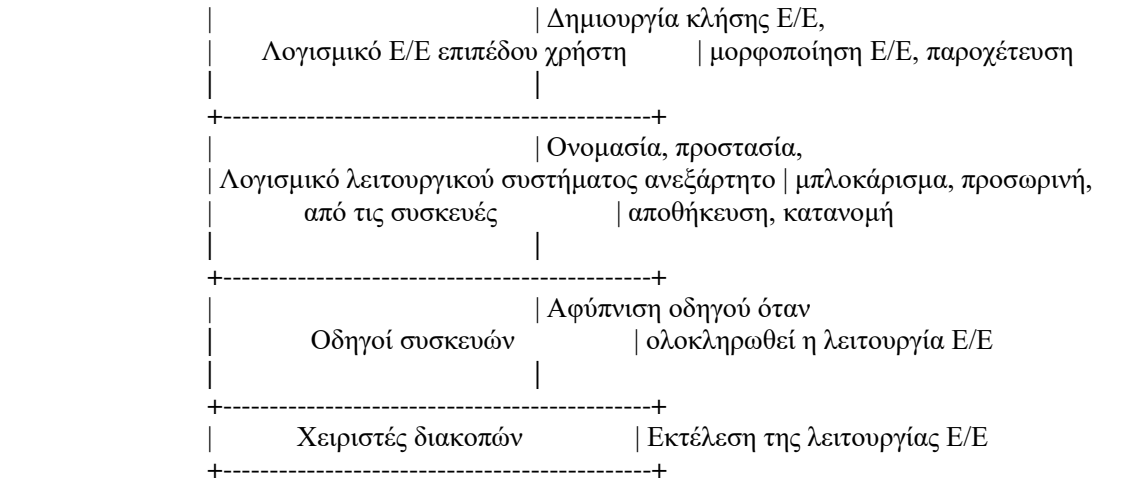
1) Μειώνεται ο αριθμός των διακοπών

Cons:

1) Ο ελεγκτής DMA είναι συνήθως πολύ πιο αργός από τη CPU

51) Ποιά τα επίπεδα λογισμικού Ε/Ε;

+----------+



52) Ποιές οι ενέργειες του χειριστή διακοπών όταν συμβαίνει μία διακοπή;

1) Αποθηκεύει όσους registers δεν έχει αποθηκεύσει το υλικό
2) Καθορίζει το περιβάλλον για τη διαδικασία εξυπηρέτησης διακοπής
3) Δημιουργεί μία στοίβα για τη διαδικασία εξυπηρέτησης διακοπής
4) Στέλνει επιβεβαίωση στον ελεγκτή διακοπών και απενεργοποιεί τις διακοπές
5) Αντιγράφει τους registers από τη στοίβα στον πίνακα διεργασιών
6) Εκτελεί τη διαδικασία εξυπηρέτησης της διακοπής
7) Επιλέγει τη διεργασία που πρόκειται να εκτελεστεί στη συνέχεια
8) Καθορίζει το περιβάλλον για τη διεργασία που θα εκτελεστεί αμέσως μετά
9) Φορτώνει τους registers της νέας διεργασίας
10) Ξεκινά την εκτέλεση της νέας διεργασίας

53) Τι είναι οι οδηγοί συσκευών (device drivers);

Κάθε συσκευή Ε/Ε συνδεδεμένη στον υπολογιστή χρειάζεται κώδικα εξειδικευμένο για τον έλεγχό της. Ο κώδικας αυτός ονομάζεται οδηγός συσκευής.

54) Τι γνωρίζετε για το μείζονα και τον ελάσσονα αριθμό συσκευής;

Μείζονας αριθμός συσκευής (major device number)
Χρησιμοποιείται για να εντοπιστεί ο κατάλληλος οδηγός συσκευής.

Ελάσσονας αριθμός συσκευής (minor device number)
Μεταβιβάζεται ως παράμετρος στον οδηγό για να προσδιορίζει τη μονάδα στην οποία πρέπει να γίνει η ανάγνωση ή η εγγραφή.

55) Τι είναι η διπλή προσωρινή αποθήκευση (double buffering)

Αποτελεί μία τεχνική Ε/Ε κατά την οποία όταν γεμίσει ένας buffer πχ κατά την είσοδο δεδομένων από ένα μόντεμ, μέχρι αντιγραφούν τα δεδομένα του buffer από τον πυρήνα στο χώρο χρήστη, νέα δεδομένα εισερχόμενα από το μόντεμ μπορούν να αποθηκεύονται σε έναν δευτερεύον buffer. Όταν και αυτός γεμίσει, επαναχρησιμοποιείται ο 1ος, αφού τώρα θα είναι άδειος. Η

διαδικασία πάει εναλλάξ.

56) Τι είναι η κυκλική προσωρινή μνήμη (circular buffer);

Αποτελείται από μία περιοχή της μνήμης και δύο δείκτες. Ο ένας δείκτης δείχνει στην επόμενη ελεύθερη λέξη, όπου μπορούν να τοποθετηθούν νέα δεδομένα. Ο άλλος δείκτης δείχνει στην πρώτη λέξη δεδομένων στην προσωρινή μνήμη, η οποία δεν έχει ακόμη αφαιρεθεί. Σε πολλές περιπτώσεις, το υλικό προωθεί το πρώτο δείκτη καθώς προσθέτει νέα δεδομένα και το λειτουργικό σύστημα προωθεί το δεύτερο δείκτη καθώς αφαιρεί και επεξεργάζεται τα δεδομένα. Και οι δύο δείκτες επιστρέφουν κυκλικά στη βάση όταν φτάσουν στην κορυφή.

57) Περιγράψτε τα επίπεδα RAID.

1) RAID 0 (Disk striping)

Διαιρεί δεδομένα σε οποιονδήποτε αριθμό δίσκων και επιτρέπει υψηλότερο throughput. Ένα μεμονωμένο αρχείο διαβάζεται από πολλαπλούς δίσκους. Αυτό το επίπεδο RAID έχει το πλεονέκτημα του υψηλού performance, ωστόσο δεν είναι fault tolerant διότι δεν κρατάει αντίγραφα των δεδομένων.

2) RAID 1 (Disk Mirroring)

Γράφει και διαβάζει ταυτόσημα δεδομένα σε ζεύγη μονάδων δίσκου. Αυτή η διαδικασία ονομάζεται συχνά mirroring δεδομένων και η κύρια λειτουργία της είναι η παροχή αντιγράφων. Εάν κάποιος από τους δίσκους στη αποτύχει, το σύστημα μπορεί ακόμα να έχει πρόσβαση στα δεδομένα από τους υπόλοιπους δίσκους.

3) RAID 5 (Striping with parity)

Διαιρεί τα αρχεία σε πολλαπλούς δίσκους, όπως το RAID 0 και επιπλέον αποθηκεύει extra πληροφορίες (μικρό μέγεθος δεδομένων που μπορούν να περιγράψουν με ακρίβεια μεγαλύτερα ποσά δεδομένων), το οποίο χρησιμοποιείται για την ανάκτηση των δεδομένων σε περίπτωση αποτυχίας δίσκου.

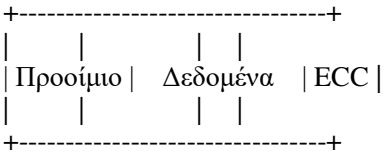
4) RAID 6 (Striping with double parity)

Το Raid 6 είναι παρόμοιο με το RAID 5, ωστόσο, παρέχει αυξημένη αξιοπιστία καθώς αποθηκεύει ένα επιπλέον πακέτο πληροφορίας.
Αυτό ουσιαστικά σημαίνει ότι είναι δυνατό για δυο δίσκους να αποτύχουν ταυτόχρονα χωρίς να υπάρξει πρόβλημα.

5) RAID 10 (Striping + Mirroring)

Το RAID 10 συνδυάζει το mirroring της RAID 1 με το striping του RAID 0.
Ή με άλλα λόγια, συνδυάζει τα αντίγραφα του RAID 1 με την αυξημένη απόδοση του RAID 0

58) Περιγράψτε τη μορφή ενός τομέα δίσκου (disk sector).



- Προοίμιο
  Μία συγκεκριμένη ακολουθία από bits που επιτρέπει στο υλικό να αναγνωρίζει την αρχή του τομέα. Περιέχει επίσης τους αριθμούς κυλίνδρου και τομέα και κάποιες άλλες πληροφορίες.

- Δεδομένα
  Το μέγεθος του τμήματος δεδομένων προσδιορίζεται από το πρόγραμμα που κάνει τη διαμόρφωση χαμηλού επιπέδου.

- ECC
  Το πεδίο ECC (error-correcting code) περιέχει πλεονάζουσες πληροφορίες οι οποίες μπορούν να χρησιμοποιηθούν κατά την ανάκαμψη από σφάλματα ανάγνωσης.

59) Ποιοί οι παράγοντες που επηρεάζουν τον χρόνο ανάγνωσης του δίσκου;

1) Χρόνος αναζήτησης (seek time - ο χρόνος που χρειάζεται για να μετακινηθεί ο βραχίονας στον κατάλληλο κύλινδρο)

2) Καθυστέρηση λόγω περιστροφής (rotational delay - ο χρόνος που χρειάζεται για να περιστραφεί ο κατάλληλος τομέας κάτω από την κεφαλή)

3) Χρόνος μεταφοράς δεδομένων

60) Περιγράψτε τους αλγόριθμους περιστροφής του βραχίονα.

1) Εξυπηρέτηση με βάση τη σειρά άφιξης (FCFS - First Come First Served)
   Ο οδηγός του δίσκου εξυπηρετεί τις αιτήσεις μία-μία, με τη σειρά που φτάνουν

2) Πρώτα η Συντομότερη Αναζήτηση (SSF - Shortest Seek First)
   Ο οδηγός εξυπηρετεί πρώτα την πιο κοντινή αίτηση, με σκοπό την ελαχιστοποίηση των κινήσεων του βραχίονα

3) Αλγόριθμος του ανελκυστήρα (Elevator algorithm)
   Ο βραχίονας κινείται προς μία κατεύθυνση μέχρι να μην υπάρχει αίτηση που εκκρεμεί προς τα εκεί. Έτσι, αλλάζει κατεύθυνση.

   Απαιτεί από το λογισμικό να διατηρεί μόνο ένα bit: το bit της κατεύθυνσης, το οποίο παίρνει τις τιμές ΠΑΝΩ και ΚΑΤΩ.

   Όταν ολοκληρωθεί η εξυπηρέτηση μιας αίτησης ο οδηγός του δίσκου ελέγχει αυτό το bit. Αν έχει τιμή ΠΑΝΩ, ο βραχίονας του δίσκου μετακινείται στην επόμενη αίτηση που εκκρεμεί και οδηγεί σε υψηλότερη θέση. Αν δεν υπάρχουν εκκρεμείς αιτήσεις για υψηλότερες θέσεις, το bit κατεύθυνσης αντιστρέφεται. Αν έχει τιμή ΚΑΤΩ, η μετακίνηση γίνεται με βάση την επόμενη αίτηση που εκκρεμεί και οδηγεί σε χαμηλότερη θέση, αν βέβαια

υπάρχει.

61) Ποιές οι λειτουργίες τις ευσταθούς αποθήκευσης;

Η ευσταθής αποθήκευση χρησιμοποιεί εάν ζευγάρι πανομοιότυπων δίσκων - μονάδων, στους οποίους τα αντίστοιχα μπλοκ συνεργάζονται ώστε να σχηματίσουν ένα μπλοκ χωρίς σφάλματα. Αν δεν υπάρχουν σφάλματα, τα αντίστοιχα μπλοκ στους δύο δίσκους είναι ίδια.

1) Ευσταθής εγγραφές (stable writes)

Αποτελείται από την εγγραφή του μπλοκ και στη συνέχεια την ανάγνωσή του από εκεί ώστε να επαληθευτεί ότι γράφτηκε σωστά. Αν δεν έχει γραφτεί σωστά, η εγγραφή και η ανάγνωση επαναλαμβάνονται n φορές μέχρι να γίνουν σωστά.

2) Ευσταθής ανάγνωση (stable reads)

Εάν προκύψει λανθασμένος κωδικός ECC, η ανάγνωση επαναλαμβάνεται μέχρι n φορές. Αν όλες οι αναγνώσεις δώσουν λάθος κωδικό ECC, διαβάζεται το αντίστοιχο μπλοκ από εφεδρική μονάδα.

3) Ανάκαμψη από κατάρευση (crash recovery)

Μετά από μία κατάρρευση ένα πρόγραμμα ανάκαμψης σαρώνει και τις δύο μονάδες και συγκρίνει τα αντίστοιχα μπλοκ. Αν και τα δύο μέρη ενός ζεύγους μπλοκ είναι πανομοιότυπα και σωστά, δεν γίνεται καμία ενέργεια. Αν το ένα από τα δύο δώσει λάθος κώδικα ECC, αντικαθίστανται από το αντίστοιχο σωστό μπλοκ. Αν σε ένα ζεύγος μπλοκ και τα δύο μέρη είναι σωστά αλλά διαφορετικά, το μπλοκ της μονάδας 1 αντικαθιστά το αντίστοιχο μπλοκ στην μονάδα 2.

62) Ποιοί είναι οι δύο τύποι πόρων;

- Προεκτοπίσιμοι

Πόροι οι οποίοι μπορούν να αποσπαστούν από μία διεργασία που τους ελέγχει χωρίς άσχημες παρενέργειες. πχ η μνήμη.

- Μη Προεκτοπίσιμοι

Πόροι που δεν μπορούν να αποσπαστούν από τη διεργασία που τους κατέχει, χωρίς να προκληθούν υπολογιστικά σφάλματα. πχ μονάδες εγγραφής CD.

Γενικά τα αδιέξοδα (deadlocks) αφορούν τους μη προεκτοπίσιμους πόρους.

63) Δώστε τον ορισμό του αδιέξοδου.

Ένα σύνολο διεργασιών βρίσκεται σε αδιέξοδο αν κάθε διεργασία του συνόλου περιμένει ένα συμβάν που μόνο μια άλλη διεργασία του συνόλου μπορεί να προκαλέσει.

64) Γράψτε τις συνθήκες για την εμφάνιση αδιεξόδου.

1) Συνθήκη αμοιβαίου αποκλεισμού (mutual exclusion condition)

Κάθε πόρος είναι είτε εκχωρημένος σε μία ακριβώς διεργασία είτε διαθέσιμος

2) Συνθήκη κράτησης και αναμονής (hold and wait condition)

Διεργασίες, δεσμεύουν την τρέχουσα στιγμή πόρους που τους εκχωρήθηκαν νωρίτερα, μπορούν να ζητήσουν και άλλους.

3) Συνθήκη μη προεκτόπισης (no preemption condition)

Πόροι που έχουν εκχωρηθεί σε μία διεργασία δεν μπορούν να αφαιρεθούν από αυτή με τη βία. Πρέπει η ίδια η διεργασία να τους αποδεσμεύσει εκούσια.

4) Συνθήκη κυκλικής αναμονής (circular wait condition)

Πρέπει να υπάρχει μια κυκλική αλυσίδα δύο ή περισσοτέρων διεργασιών, κάθε μία από τις οποίες περιμένει έναν πόρο που κατέχει το επόμενο μέλος της αλυσίδας.

65) Ποιοί είναι οι τρόποι ανάκαμψης από αδιέξοδο;

1) Ανάκαμψη μέσω προεκτόπισης
2) Ανάκαμψη μέσω ανασκευής (rollback)
3) Ανάκαμψη μέσω εξάλειψης διεργασιών

66) Τι είναι το κλείδωμα σε δύο φάσεις (two-phase locking);

Αποτελεί μία τεχνική για την αποφυγή αδιέξοδου. Υπάρχουν δύο διακριτές φάσεις:

1. Η διεργασία προσπαθεί να κλειδώσει όλους του πόρους που χρειάζεται, ένα κάθε φορά. Αν επιτύχει, ξεκινάει τη δεύτερη φάση, αλλιώς η διεργασία απλώς απενεργοποιεί όλα τα κλειδώματα της και ξεκινάει την πρώτη φάση από την αρχή.

2. Στη δεύτερη φάση, όπου η διεργασία έχει καταφέρει να κλειδώσει όλους τους πόρους που χρειάζεται, εκτελεί τις όποιες ενημερώσεις, και απενεργοποιεί τα κλειδώματα.

Δεν είναι γενικά εφαρμόσιμη τεχνική. Χρησιμοποιείται συνήθως για βιομηχανικές, χημικές, κτλ διεργασίες και όχι για διεργασίες που έχουμε στα λειτουργικά συστήματα.

67) Τι είναι η ομοιόμορφη προσπέλαση μνήμης (UMA - Uniform Memory Access);

Είναι η ιδιότητα που έχουν ορισμένοι πολυεπεξεργαστές πως κάθε λέξη μνήμης μπορεί να διαβαστεί με την ίδια ταχύτητα όπως οποιαδήποτε άλλη.

68) Ποιοί οι τρόποι διασύνδεσης πολυεπεξεργαστών UMA;

1) Με δίαυλο

Pros:

1) Είναι απλός

Cons:

1) Εάν ο δίαυλος είναι απασχολημένος τη στιγμή που μία CPU θέλει να διαβάσει ή να γράψει στη μνήμη, η CPU απλώς περιμένει μέχρι ο δίαυλος να τελειώσει.

Με 2 ή 3 CPU, ο ανταγωνισμός για το δίαυλο είναι διαχειρίσιμος, με 32 ή 64 CPU ο ανταγωνισμός γίνεται αφόρητος. Το σύστημα θα περιορίζεται πλήρως από το εύρος ζώνης του διαύλου, και οι περισσότερες CPU θα είναι αδρανείς το μεγαλύτερο χρονικό διάστημα.
Θα μπορούσαμε να προσθέσουμε κρυφή μνήμη για να ανεχτούμε και περισσότερες CPU αλλά και πάλι αυτή η διασύνδεση δεν κλιμακώνεται καλά.

2) Με σταυρωτό δίκτυο διασύνδεσης

Pros:

1) Αποτελεί μη ανασταλτικό δίκτυο (nonblocking network) σε αντίθεση με τη διασύνδεση μέσω διαύλου.

Cons:

1) Ο αριθμός των διασταυρώσεων αυξάνεται με το n^2

3) Με δίκτυα μεταγωγής πολλών σταδίων

Ένας μεταγωγός, πχ 2x2 έχει 2 εισόδους και 2 εξόδους. Τα μηνύματα που φτάνουν σε οποιαδήποτε γραμμή εισόδου μπορούν να μεταφερθούν σε οποιαδήποτε γραμμή εξόδου.

Pros:

1) Στη γενική περίπτωση , αν υπάρχουν n CPU και n μνήμες χρειαζόμαστε log2n στάδια, με n/2 μεταγωγούς ανά στάδιο, οπότε συνολικά χρειάζονται (n/2)log2n μεταγωγοί, αριθμός που είναι πολύ καλύτερος από τις n^2 διασταυρώσεις ειδικά για μεγάλες τιμές του n.

Cons:

1) Αποτελεί ανασταλτικό δίκτυο (blocking network)

69) Τι γνωρίζετε για την μη ομοιόμορφη προσπέλαση μνήμης (NUMA - Non Uniform Memory Access);

Οι πολυεπεξεργαστές UMA με ένα δίαυλο γενικά περιορίζονται στη χρήση το πολύ μερικών δεκάδων CPU, ενώ οι πολυεπεξεργαστές με σταυρωτά δίκτυα διασύνδεσης χρειάζονται αρκετό (ακριβό) υλικό και δεν είναι και τόσο μεγαλύτεροι. Η παρατήρηση αυτή οδηγεί στους πολυεπεξεργαστές NUMA.

Όπως και οι UMA, παρέχουν ένα μοναδικό χώρο διευθύνσεων σε όλες τις CPU αλλά, σε αντίθεση με τους UMA, η πρόσβαση στις τοπικές υπομονάδες μνήμης είναι ταχύτερη από την πρόσβαση στις απομακρυσμένες. Επομένως, όλα τα προγράμματα UMA μπορούν να εκτελούνται χωρίς αλλαγή σε μηχανές NUMA, αλλά η απόδοση θα είναι χειρότερη σε σχέση με τις μηχανές UMA που έχουν το ίδιο ρυθμό ρολογιού.

Οι μηχανές NUMA έχουν τρία κρίσιμα γνωρίσματα:

1) Υπάρχει ένας και μοναδικός χώρος διευθύνσεων ορατός σε όλες τις CPU
2) Η πρόσβαση σε απομακρυσμένες μνήμες γίνεται μέσω εντολών LOAD και STORE
3) Η πρόσβαση στις απομακρυσμένες μνήμες γίνεται βραδύτερα από την πρόσβαση στην τοπική μνήμη

70) Τι είναι η αδιακρισία (snooping);

Ειδικά κυκλώματα υλικού εξασφαλίζουν ότι, αν μια λέξη υπάρχει σε δύο ή περισσότερες κρυφές μνήμες και κάποια από τις CPU την τροποποιήσει, η λέξη αφαιρείται αυτόματα και αδιαίρετα από όλες τις κρυφές μνήμες, ώστε να διατηρηθεί η συνέπεια. Η διαδικασία αυτή είναι γνωστή ως αδιακρισία.

71) Περιγράψτε τους τύπους λειτουργικών συστημάτων για πολυεπεξεργαστές.

1) Κάθε CPU διαθέτει το δικό της λειτουργικό σύστημα

Διαιρείται η μνήμη σε στατικά σε ισάριθμα με τις CPU διαμερίσματα και δίνεται σε κάθε CPU η δική της ιδιωτική μνήμη και το δικό της ιδιωτικό αντίγραφο του λειτουργικού συστήματος. Ουσιαστικά, οι n CPU λειτουργούν σαν n ανεξάρτητοι υπολογιστές.

Pros:

1) Το μοντέλο αυτό είναι απλό στην κατασκευή.

Cons:

1) Δεν υπάρχει κοινοχρησία διεργασιών. Αν κάποιος χρήστης συνδεθεί στη CPU 1, όλες οι διεργασίες του θα εκτελούνται στη CPU 1. Συνεπώς, είναι πιθανό η CPU 1 να είναι αδρανής ενώ η CPU 2 να έχει μεγάλο φόρτο εργασίας.

2) Δεν υπάρχει κοινοχρησία σελίδων. Είναι πιθανόν η CPU 1 να έχει αρκετές διαθέσιμες σελίδες ενώ η CPU 2 να σελιδοποιεί συνεχώς. Δεν υπάρχει τρόπος η CPU 2 να δανειστεί μερικές σελίδες από τη CPUT 1, επειδή η κατανομή μνήμης είναι σταθερή.

3) Αν το λειτουργικό σύστημα διατηρεί μια προσωρινή κρυφή μνήμη για τα μπλοκ δίσκου που έχουν χρησιμοποιηθεί πρόσφατα, κάθε λειτουργικό σύστημα εκτελεί αυτή τη εργασία ανεξάρτητα από τα υπόλοιπα. Επομένως, ένα συγκεκριμένο μπλοκ δίσκου μπορεί να βρίσκεται σε πολλές προσωρινές κρυφές μνήμες ταυτόχρονα και να είναι "βρώμικο", γεγονός που οδηγεί σε ασυνεπή αποτελέσματα. Ο μόνος τρόπος να αποφευχθεί αυτό το πρόβλημα είναι να μη χρησιμοποιούνται προσωρινές κρυφές μνήμες.

Αυτό δεν είναι δύσκολο αλλά μειώνει αρκετά την απόδοση.

2) Πολυεπεξεργαστές κυρίου-υπηρέτη (master - slave)

Υπάρχει ένα αντίγραφο του λειτουργικού συστήματος και των πινάκων του στη CPU 1 και όχι στις υπόλοιπες. Όλες οι κλήσεις συστήματος ανακατευθύνονται για επεξεργασία στη CPU 1. Η CPU 1 μπορεί επίσης να εκτελεί διεργασίες χρηστών αν της μένει χρόνος.

Pros:

1) Κοινοχρησία διεργασιών. Όταν μία CPU είναι αδρανής, ζητάει από το λειτουργικό σύστημα να της αναθέσει μία διεργασία, και αφού την παραλάβει την εκτελεί. Επομένως, ποτέ μία CPU δεν είναι αδρανής την ώρα που κάποια άλλη έχει υπερφορτωθεί (καλύτερο load-balancing).

2) Κοινοχρήσια σελίδων. Οι σελίδες μπορούν να κατανεμηθούν δυναμικά στις διεργασίες.

3) Υπάρχει μόνο μία προσωρινή κρυφή μνήμη, οπότε δεν μπορούν α συμβούν ασυνέπειες.

Cons:

1) Όταν υπάρχουν πολλές CPU, μπορεί να δημιουργηθεί συμφόρηση στην κύρια, μιας και αυτή θα πρέπει να χειριστεί όλες τις κλήσεις συστήματος που προέρχονται από τις υπόλοιπες.

3) Συμμετρικοί πολυεπεξεργαστές (SMP - Symmetric Multiprocessor)

Υπάρχει ένα αντίγραφο του λειτουργικού συστήματος στη μνήμη, αλλά μπορεί να το εκτελέσει οποιαδήποτε CPU. Όταν γίνεται μία κλήση συστήματος, η CPU που την έκανε μεταφέρεται σε κατάσταση λειτουργίας πυρήνα και επεξεργάζεται την κλήση συστήματος.

Pros:

1) Εξισορροπεί δυναμικά τις διεργασίες και τη μνήμη
2) Εξαλείφει επίσης τη συμφόρηση στη κύρια CPU αφού δεν υπάρχει τέτοια

Cons:

1) Επειδή το λειτουργικό σύστημα είναι 1 και το εκτελούν όλες οι CPU, απαιτείται συγχρονισμός, πράγμα που κάνει τον κώδικα του λειτουργικού πιο περίπλοκο. Πρέπει να βρεθούν οι κρίσιμες περιοχές και να προστατευτούν με mutexes.

72) Σχολιάστε σχετικά με περιστροφή vs εναλλαγή.

Εάν κάποιο νήμα που εκτελείται σε μία CPU χρειάζεται να προσπελάσει την προσωρινή κρυφή μνήμη του συστήματος αρχείων και αυτή είναι κλειδωμένη, η CPU μπορεί να αποφασίσει να περιμένει με περιστροφή ή να κάνει εναλλαγή σε ένα διαφορετικό νήμα αντί να περιμένει.

Ο συμβιβασμός είναι ο εξής: η περιστροφή σπαταλάει με άμεσο τρόπο κύκλους της CPU. Η επαναλαμβανόμενη δοκιμή ενός κλειδώματος δεν είναι παραγωγική εργασία. Η εναλλαγή πάντως σπαταλάει επίσης κύκλους CPU εφόσον πρέπει να αποθηκευτεί η κατάσταση του τρέχοντος νήματος, να αποκτηθεί το κλείδωμα της λίστας των έτοιμων διεργασιών, να επιλεγεί ένα νήμα, να φορτωθεί η κατάστασή του, και να ξεκινήσει η εκτέλεσή του. Επιπλέον, η κρυφή μνήμη της CPU θα περιέχει όλα τα λανθασμένα μπλοκ όποτε, κατά την εκκίνηση του νέου νήματος, θα συμβούν αρκετές χρονοβόρες αστοχίες κρυφής μνήμης. Είναι πολύ πιθανό να συμβούν επίσης αστοχίες TLB. Τέλος, πρέπει να γίνει επιστροφή στο αρχικό νήμα η οποία θα οδηγήσει σε πρόσθετες αστοχίες κρυφής μνήμης. Θα χαθούν οι κύκλοι για τις δύο θεματικές εναλλαγές και για τις αστοχίες της κρυφής μνήμης.

Το πρόβλημα είναι πως οι κρίσιμες περιοχές ποικίλουν σε μεγάλο βαθμό ως προς τη διάρκειά τους, οπότε ποιά προσέγγιση είναι η καλύτερη;

Τα καλύτερα αποτελέσματα επιτυγχάνονται όταν το σύστημα καταγράφει έναν αριθμό από τους τελευταίους χρόνους περιστροφής και υποθέτει ότι η συγκεκριμένη περίπτωση θα είναι παρόμοια με τις προηγούμενες.

Με άλλα λόγια με τη χρήση ενός αλγορίθμου, θα γίνεται η υπόθεση για το χρόνο περιστροφής, εάν αυτός είναι μικρότερος από το πιθανό κόστος εναλλαγής, η CPU περιστρέφεται, αλλιώς, εναλλάσσει το νήμα με ένα άλλο.

73) Ποιοί οι τρόποι χρονο προγραμματισμού πολυεπεξεργαστών;

1) Χρονομερισμός

Υπάρχει μόνο μία δομή δεδομένων σε όλο το σύστημα για τα έτοιμα νήματα, πιθανόν όπως μια λίστα ή ακόμη καλύτερα, ένα σύνολο από λίστες για τα νήματα με διαφορετικές προτεραιότητες.

Pros:

1) Απλός στην υλοποίηση

2) Αυτόματη εξισορρόπηση φορτίου επειδή δεν μπορεί ποτέ μία CPU να είναι αδρανής και οι άλλες υπερφορτωμένες.

Cons:

1) Ανταγωνισμός για τη δομή δεδομένων χρονο προγραμματισμού καθώς αυξάνεται ο αριθμός των CPU

2) Η συνήθης επιβάρυνση της θεματικής εναλλαγής κατά το μπλοκάρισμα ενός νήματος για Ε/Ε

2) Χωρομερισμός (space sharing)

Αν τα νήματα μιας διεργασίας επικοινωνούν πολύ, είναι χρήσιμο να εκτελούνται ταυτόχρονα. Ο ταυτόχρονος χρονο προγραμματισμός πολλών νημάτων σε πολλές CPU ονομάζεται χωρομερισμός (space sharing)

Υποθέστε πως μία ολόκληρη ομάδα από σχετιζόμενα νήματα δημιουργείται ταυτόχρονα. Τη στιγμή της δημιουργίας τους, χρονο προγραμματιστής ελέγχει αν υπάρχουν ισάριθμες ελεύθερες CPU. Αν υπάρχουν, σε κάθε νήμα ανατίθεται η αποκλειστική του CPU, και ξεκινούν όλα μαζί. Αν δεν υπάρχουν αρκετές CPU, δεν ξεκινάει κανένα από τα νήματα μέχρι να γίνει διαθέσιμος ο αντίστοιχος αριθμός CPU.

Pros:

1) Εξάλειψη της επιβάρυνσης λόγω θεματικών εναλλαγών

Cons:

1) Χάνεται χρόνος όταν κάποια CPU μπλοκάρεται και δεν κάνει τίποτε άλλο μέχρι να έλθει ξανά σε ετοιμότητα.

3) Χρονοπρογραμματισμός ομάδας (gang scheduling)

Αποτελείται από 3 τμήματα:

1) Οι ομάδες σχετιζόμενων νημάτων χρονο προγραμματίζονται ως μία ενότητα.

2) Όλα τα μέλη μιας ομάδας εκτελούνται ταυτόχρονα,σε διαφορετικές χρονομεριζόμενες CPU.

3) Όλα τα μέλη της ομάδας ξεκινούν και τερματίζουν τα χρονομερίδιά τους ταυτόχρονα.

Ο χρόνος διαιρείται σε διακριτά κβάντα. Στην αρχή κάθε νέου κβάντου, επαναχρονοπρογραμματίζονται όλες οι CPU και σε κάθε μία από αυτές ξεκινάει ένα νέο νήμα. Στην αρχή του επόμενου κβάντου, ξαναγίνεται χρονο προγραμματισμός. Στο μεσοδιάστημα, δε γίνεται καθόλου χρονο προγραμματισμός. Αν κάποιο νήμα μπλοκαριστεί, η αντίστοιχη CPU παραμένει αδρανής μέχρι το τέλος του κβάντου.

Pros:

1) Η βασική ιδέα είναι πως εκτελούνται μαζί όλα τα νήματα μιας διεργασίας ώστε, αν κάποιο από αυτά στείλει μια αίτηση σε κάποιο άλλο, το δεύτερο να παραλάβει σχεδόν αμέσως το μήνυμα και να έχει τη δυνατότητα να απαντήσει σχεδόν στιγμιαία.

74) Τι είναι η δρομολόγηση συγγένειας (affinity scheduling);

Είναι μία τεχνική που εξασφαλίζει ότι μία διεργασία ή ένα thread μετά από κάποια εναλλαγή θα συνεχίσει να τρέχει στην ίδια CPU.

Pros:
- Μείωση των cache misses
  Η scheduler αφού εναλλάξει το process, θα το επαναφέρει στην ίδια CPU και ό,τι δεδομένα είχε η CPU στην cache της εξακολουθεί να τα

χρησιμοποιεί. Εάν το process εκτελούνταν σε άλλη CPU μετά την εναλλαγή, εκείνη θα έπρεπε να φέρει στην cache της τα δεδομένα του, προκαλώντας τόσο cache miss στην ίδια όσο και cache invalidation στην προηγούμενη (εάν πείραζε τα δεδομένα)

75) Τι γνωρίζετε για τους πολυ-υπολογιστές (multicomputers);

Αποτελούνται από συζευγμένες CPU που δε μοιράζονται μνήμη. Κάθε μία έχει τη δική της μνήμη. Το μυστικό της υψηλής απόδοσης είναι η έξυπνη σχεδίαση του δικτύου σύνδεσης και της κάρτας διασύνδεσης. Ο στόχος είναι η αποστολή μηνυμάτων σε κλίμακα μικροδευτερολέπτου, και όχι η προσπέλαση μνήμης σε κλίμακα νανοδευτερολέπτου.

Κάθε κόμβος διαθέτει μία κάρτα διασύνδεσης δικτύου από την οποία βγαίνουν ένα ή δύο καλώδια. Τα καλώδια αυτά συνδέονται είτε με άλλους κόμβους είτε με μεταγωγούς (switches). Υπάρχουν διάφορες τοπολογίες διασύνδεσης:

1) Απλός μεταγωγός
2) Δακτύλιος
3) Πλέγμα
4) Διπλό πλέγμα
5) Κύβος
6) Τετραδιάστατος υπερκύβος

76) Τι γνωρίζετε για την εικονικοποίηση (virtualization);

Είναι μία τεχνολογία που επιτρέπει σε ένα μοναδικό υπολογιστή να φιλοξενεί πολλές εικονικές μηχανές, που κάθε μία τους ίσως χρησιμοποιεί διαφορετικό λειτουργικό σύστημα.

Pros:

1) Μία βλάβη σε μία εικονική μηχανή δε συμπαρασύρει και τις υπόλοιπες αυτόματα (απομόνωση - isolation)

2) Εξοικονόμηση χώρου και κόστους

3) Εκτέλεση παλαιότερων εφαρμογών σε λειτουργικά συστήματα που δεν υποστηρίζονται πλέον ή που δεν συνεργάζονται με το τρέχον υλικό.

4) Ευκολότερη ανάπτυξη λογισμικού για διαφορετικά λειτουργικά συστήματα

Cons:

1) Βλάβη στον διακομιστή που φιλοξενεί τις εικονικές μηχανές, έχει καταστροφικό αποτέλεσμα

77) Τι γνωρίζετε για τους υπερεπόπτες τύπου 1 και 2 (hypervisors type 1, 2);

- Υπερεπόπτης τύπου 1

Στην πραγματικότητα πρόκειται για το λειτουργικό σύστημα, αφού είναι το μοναδικό πρόγραμμα που εκτελείται σε κατάσταση πυρήνα. Καθήκον του είναι

η υποστήριξη πολλών αντιγράφων του πραγματικού υλικού, που ονομάζονται εικονικές μηχανές (virual machines) και είναι παρόμοιες με τις διεργασίες που υποστηρίζει ένα κανονικό λειτουργικό σύστημα.

Αν το λειτουργικό σύστημα επισκέπτη εκτελέσει μία ευαίσθητη εντολή, συμβαίνει μια παγίδευση προς τον πυρήνα. Στη συνέχεια, ο υπερεπόπτης μπορεί να επιθεωρήσει την εντολή για να διαπιστώσει αν δόθηκε από το λειτουργικό σύστημα επισκέπτη της εικονικής μηχανής ή από κάποιο πρόγραμμα χρήστη της εικονικής μηχανής. Στην πρώτη περίπτωση, επιτρέπει την εκτέλεση της εντολής, στη δεύτερη περίπτωση, εξομοιώνει την αντίδραση του πραγματικού υλικού όταν αντιμετωπίζει μια ευαίσθητη εντολή που εκτελείται σε κατάσταση χρήστη.

- Υπερεπόπτης τύπου 2

Πρόκειται απλώς για ένα πρόγραμμα χρήστη που εκτελείται, για παράδειγμα, στα Windows ή το Linux και "ερμηνεύει" το σύνολο εντολών της μηχανής, το οποίο δημιουργεί και αυτό μια εικονική μηχανή.

Όλες οι ευαίσθητες εντολές αντικαθίστανται, από κλήσεις διαδικασιών που εξομοιώνουν τις εντολές αυτές. Ποτέ δεν εκτελούνται ευαίσθητες εντολές του λειτουργικού συστήματος επισκέπτη από το πραγματικό υλικό. Μετατρέπονται σε κλήσεις προς τον υπερεπόπτη, οποίος στη συνέχεια τις εξομοιώνει.

Το λειτουργικό σύστημα που εκτελείται επάνω στον υπερεπόπτη και στις δύο περιπτώσεις ονομάζεται λειτουργικό σύστημα επισκέπτη (guest operating system)
Στην περίπτωση του υπερεπόπτη τύπου 2, το λειτουργικό σύστημα που εκτελείται επάνω στο υλικό ονομάζεται λειτουργικό σύστημα υπηρεσίας (host operating system)

Στην περίπτωση του υπερεπόπτη τύπου 2, δεν χρειάζεται να υποστηρίζεται εικονικοποίηση από τη CPU.

Είναι καθήκον του υπερεπόπτη να δίνει την ψευδαίσθηση, και μάλιστα αποδοτικά, πως οι εικονικές μηχανές είναι πραγματικές (τρέχουν πάνω από το υλικό).

78) Τι είναι η παρα-εικονικοποίηση (paravirtualization);

Είναι μία τεχνική κατά την οποία το λειτουργικό σύστημα επισκέπτη, αντί να εκτελεί ευαίσθητες εντολές, κάνει κλήσεις στον υπερεπόπτη. Στην πράξη, το λειτουργικό σύστημα επισκέπτης συμπεριφέρεται σαν ένα πρόγραμμα χρήστη που κάνει κλήσεις στο λειτουργικό σύστημα (τον υπερεπόπτη).

Ο υπερεπόπτης λειτουργεί σαν ένας μικροπυρήνας.

Η εξομοίωση περίεργων εντολών υλικού είναι μία δυσάρεστη και χρονοβόρα εργασία. Απαιτεί μια κλήση προς τον υπερεπόπτη και, στη συνέχεια, την εξομοίωση της ακριβούς σημασιολογίας μιας πολύπλοκης εντολής. Είναι πολύ καλύτερα απλώς να αφήσουμε το λειτουργικό σύστημα επισκέπτη να καλέσει τον υπερεπόπτη για Ε/Ε, κ.ο.κ.

Είναι απαραίτητο το λειτουργικό σύστημα επισκέπτη να έχει μεταγλωττιστεί με την υποστήριξη για παρα-εικονικοποίηση.

79) Ποιές οι διαφορές πολυεπεξεργαστών, πολυ-υπολογιστών και κατανεμημένων συστημάτων;

Στοιχείο	Πολυεπεξεργαστής	Πολυ-υπολογιστής	Κατανεμ. Σύστημα
Διευθέτηση κόμβων	CPU	CPU, RAM διασύνδεση δικτύου	Πλήρης υπολογιστής
Περιφερειακά κόμβου	Όλα κοινόχρηστα	Κοινή χρήση εκτός ίσως του δίσκου	Πλήρες σύνολο ανά κόμβο
Θέση	Ίδιο πλαίσιο (rack)	Ίδιο δωμάτιο	Σε ολόκληρο τον πλανήτη
Επικοινωνία κόμβων	Κοινόχρηστη RAM	Αποκλειστική διασύνδεση	Παραδοσιακό δίκτυο
Λειτουργικά συστήματα	Ένα, κοινόχρηστο	Πολλά, ίδια	Πιθανόν όλα διαφορετικά
Συστήματα αρχείων	Ένα, κοινόχρηστο	Ένα, κοινόχρηστο	Κάθε κόμβος το δικό του
Διαχείριση	Ένας οργανισμός	Ένας οργανισμός	Πολλοί οργανισμοί

80) Ποιά τα βήματα του φυλλομετρητή για να φέρει τη σελίδα http://www.minix3.org/doc/faw.html;

1. Ο φυλλομετρητής ζητάει από το σύστημα DNS τη διεύθυνση IP του www.minix3.org
2. Το DNS απαντάει ότι η διεύθυνση αυτή είναι 130.37.20.20
3. Ο φυλλομετρητής δημιουργεί με τη διεύθυνση 130.37.20.20 μια σύνδεση TCP στη θύρα 80
4. Ο φυλλομετρητής στέλνει μια αίτηση ζητώντας το αρχείο doc/faq.html
5. Διακόπτεται η σύνδεση TCP
6. Ο φυλλομετρητής εμφανίζει όλο το κείμενο του αρχείου doc/faq.html
7. Ο φυλλομετρητής προσκομίζει και εμφανίζει όλες τις εικόνες του αρχείου doc/faq.html

81) Ποιοί είναι οι στόχοι της ασφάλειας λειτουργικών συστημάτων;

1) Εμπιστευτικότητα δεδομένων
2) Ακεραιότητα δεδομένων
3) Διαθεσιμότητα συστήματος
4) Αποκλεισμός εισβολέων

82) Ποιά είναι η Αρχή του Kerckhoffs (Kerckhoffs' Principle);

Οι αλγόριθμοι πρέπει να είναι όλοι δημόσιοι και η μυστικότητα να βρίσκεται αποκλειστικά στα κλειδιά.

83) Τι είναι η κρυπτογραφία μυστικού κλειδιού (secret key cryptography) ή κρυπτογραφία συμμετρικού κλειδιού (symmetric key cryptography);

Αποτελεί είδος κρυπτογράφησης, όπου εάν δοθεί το κλειδί κρυπτογράφησης, είναι εύκολο να βρεθεί το κλειδί αποκρυπτογράφησης.
Το κλειδί είναι ένα, και όπως λέει και ο τίτλος, είναι μυστικό.

Pros:

1) Είναι αποδοτική επειδή η ποσότητα υπολογισμών που απαιτείται για να κρυπτογραφηθεί ή να αποκρυπτογραφηθεί ένα μήνυμα είναι διαχειρίσιμη.

Cons:

1) Ο αποστολέας και ο παραλήπτης πρέπει να γνωρίζουν το κοινό μυστικό κλειδί. Ίσως πρέπει να συναντηθούν προσωπικά για να δώσει ο ένας το κλειδί στον άλλο.

84) Τι είναι η κρυπτογραφία δημόσιου κλειδιού (public key cryptography);

Εδώ χρησιμοποιούνται διαφορετικά κλειδιά για την κρυπτογράφηση και την αποκρυπτογράφηση και, αν δοθεί ένα καλά επιλεγμένο κλειδί κρυπτογράφησης, είναι πρακτικά αδύνατον να ανακαλυφθεί το αντίστοιχο κλειδί αποκρυπτογράφησης. Κάτω από αυτές τις περιστάσεις, το κλειδί κρυπτογράφησης μπορεί να δημοσιοποιηθεί και να διατηρείται μυστικό μόνο το κλειδί αποκρυπτογράφησης.
Ο τρόπος λειτουργίας της κρυπτογραφίας δημόσιου κλειδιού είναι ότι και τα δύο μέρη διαλέγουν ένα ζεύγος (δημόσιο κλειδί, ιδιωτικό κλειδί) και δημοσιοποιούν το δημόσιο κλειδί. Το δημόσιο κλειδί είναι το κλειδί κρυπτογράφησης, το ιδιωτικό κλειδί είναι το κλειδί αποκρυπτογράφησης.

Pros:

1) Δεν υπάρχει το πρόβλημα αποστολής ενός μοναδικού, μυστικού κλειδιού μεταξύ αποστολέα και παραλήπτη
2) Υπολογιστικά δύσκολο να βρεθεί το μυστικό κλειδί αποκρυπτογράφησης

Cons:

1) Είναι χίλιες φορές πιο αργή από τη συμμετρική κρυπτογραφία

85) Τι γνωρίζετε για τις ψηφιακές υπογραφές;

Μια ψηφιακή υπογραφή είναι ένα μαθηματικό σχήμα για την επαλήθευση της αυθεντικότητας των ψηφιακών μηνυμάτων ή εγγράφων.
Μια έγκυρη ψηφιακή υπογραφή δίνει στον παραλήπτη το λόγο να πιστεύει ότι το μήνυμα δημιουργήθηκε από έναν γνωστό αποστολέα (έλεγχος ταυτότητας), ότι ο αποστολέας δεν μπορεί να αρνηθεί την αποστολή του μηνύματος (μη αποκήρυξη) και ότι το μήνυμα δεν μεταβλήθηκε κατά τη μεταφορά (ακεραιότητα).

86) Τι εννοούμε με τον όρο τομέας (domain) και δικαίωμα (right);

Ο τομέας είναι ένα σύνολο ζευγών (αντικείμενο, δικαιώματα). Κάθε ζεύγος προσδιορίζει ένα αντικείμενο και κάποιο υποσύνολο των λειτουργιών που μπορούν να εκτελεστούν σε αυτό. Δικαίωμα με αυτή την έννοια σημαίνει άδεια εκτέλεσης μιας από τις λειτουργίες.

87) Τι είναι ένα μητρώο προστασίας (protection matrix);

Είναι ένας μεγάλος δισδιάστατος πίνακας όπου οι γραμμές αντιστοιχούν στους τομείς και οι στήλες στα αντικείμενα. Κάθε κελί περιέχει τα δικαιώματα, αν υπάρχουν, που διαθέτει ο τομέας για το συγκεκριμένο αντικείμενο.

	Αρχείο 1	Αρχείο 2	Αρχείο 3	Αρχείο 4
Τομέας 1	R	RW		
Τομέας 2	W		X	W
Τομέας 3		W	RWX	R

Στη πράξη η αποθήκευση του μητρώου γίνεται σπάνια επειδή είναι μεγάλο και αραιό. Τα περισσότερα πεδία δεν έχουν καθόλου πρόσβαση στα περισσότερα αντικείμενα, οπότε η αποθήκευση ενός πολύ μεγάλου και κατά το μεγαλύτερο μέρος κενού μητρώου είναι σπατάλη χώρου δίσκου.
Δύο μέθοδοι που είναι όμως πρακτικές είναι η αποθήκευση του μητρώου κατά γραμμές ή κατά στήλες, και στη συνέχεια η αποθήκευση μόνο των μη κενών στοιχείων.

88) Τι είναι οι λίστες έλεγχου πρόσβασης (ACL - Access Control Lists);

Μία τεχνική που αντιστοιχίζει σε κάθε αντικείμενο μία ταξινομημένη λίστα, η οποία περιέχει όλους τους τομείς που μπορούν να το προσπελάσουν καθώς και τον επιτρεπόμενο τρόπο προσπέλασης.

F1	A: RW; B: R
F2	A: R; B: RW; C: R
F3	B: RWX; C: RX

- F1, F2, F3 είναι αρχεία - αντικείμενα
- A, B, C είναι τομείς / χρήστες
- R, W, X είναι τα δικαιώματα (read, write, execute)

89) Ποιοί είναι οι 3 γενικοί κανόνες, σύμφωνα με τους οποίους προσδιορίζεται η ταυτότητα:

1) Κάτι το οποίο γνωρίζει ο χρήστης

2) Κάτι το οποίο κατέχει ο χρήστης
3) Κάτι το οποίο είναι ο χρήστης

90) Αναφέρετε τεχνικές πιστοποίησης ταυτότητας.

1) Πιστοποίηση με ρήση κωδικών πρόσβασης
2) Πιστοποίηση με χρήση φυσικού αντικειμένου
3) Πιστοποίηση με χρήση βιομετρίας

91) Για την ασφάλεια των κωδικών πρόσβασης στο UNIX, χρησιμοποιείται "αλάτι" (salt). Τι γνωρίζεται γι' αυτό;

Η ιδέα είναι να αντιστοιχιστεί σε κάθε κωδικό πρόσβασης ένας τυχαίος αριθμός των n bit, ο οποίος ονομάζεται αλάτι (salt). Ο τυχαίος αυτός αριθμός αλλάζει όποτε αλλάξει και ο κωδικός. Ο τυχαίος αριθμός αποθηκεύεται στο αρχείο κωδικών πρόσβασης σε μη κρυπτογραφημένη μορφή, ώστε ο καθένας να μπορεί να τον διαβάσει. Αντί να αποθηκεύεται απλώς ο κρυπτογραφημένος κωδικός πρόσβασης στο αρχείο κωδικών πρόσβασης, πρώτα συνενώνονται και κρυπτογραφούνται μαζί ο κωδικός πρόσβασης και ο τυχαίος αριθμός. Αυτό το κρυπτογραφημένο αποτέλεσμα αποθηκεύεται στο αρχείο κωδικών πρόσβασης. Κάθε χρήστης έχει μία γραμμή του αρχείου, με τρεις καταχωρίσεις που χωρίζονται με κόμματα: όνομα χρήστη, αλάτι, και κρυπτογραφημένο το συνδυασμό κωδικός πρόσβασης + αλάτι.

Τώρα, έστω πως ένας επιτιθέμενος θέλει να κατασκευάσει μία λίστα από πιθανούς κωδικού πρόσβασης, να τους κρυπτογραφήσει, και να αποθηκεύσει τα αποτελέσματα σε ένα ταξινομημένο αρχείο f, ώστε να μπορεί να αναζητήσει οποιοδήποτε κρυπτογραφημένο κωδικό πρόσβασης εύκολα. Αν ο επιτιθέμενος υποπτευθεί ότι το PASS μπορεί να είναι κωδικός πρόσβασης, δεν αρκεί πλέον να κρυπτογραφήσει το PASS και να τοποθετήσει το αποτέλεσμα στο f. Πρέπει να κρυπτογραφήσει 2^n συμβολοσειρές όπως οι PASS0000, PASS0001, PASS0002, κ.ο.κ. και να τις τοποθετήσει στο f. Η τεχνική αυτή αυξάνει το μέγεθος του f κατά 2^n. Το UNIX χρησιμοποιεί αυτή τη μέθοδο με n=12.

92) Τι γνωρίζεται για επιθέσεις από το εσωτερικό του συστήματος;

Τις φέρνουν σε πέρα προγραμματιστές ή άλλοι εργαζόμενοι της εταιρείας που δουλεύουν στον προς προστασία υπολογιστή ή κατασκευάζουν κρίσιμο λογισμικό.

1) Λογικές βόμβες (logic bombs)

Είναι ένα τμήμα κώδικα που έχει γραφεί από κάποιο προγραμματιστή (οποίος δουλεύει ακόμη στην εταιρεία) και έχει τοποθετηθεί κρυφά στο σύστημα παραγωγής. Όσο ο προγραμματιστής τροφοδοτεί τον κώδικα με τον ημερήσιο κωδικό πρόσβασης του, δεν συμβαίνει τίποτε. Αν όμως ο προγραμματιστής απολυθεί ξαφνικά την επόμενη ημέρα (ή εβδομάδα), η λογική βόμβα δεν παραλαμβάνει τον ημερήσιο κωδικό πρόσβασης και "εκρήγνυται". Η "έκρηξη" της βόμβας μπορεί να προκαλέσει καθαρισμό του δίσκου, διαγραφή τυχαίων αρχείων, δύσκολες να εντοπιστούν αλλαγές σε κρίσιμα προγράμματα, ή κρυπτογράφηση βασικών αρχείων.

2) Καταπακτές (Trap doors)

Το πρόβλημα αυτό αυτό δημιουργείται μέσω κώδικα που εισάγεται στο σύστημα από κάποιο προγραμματιστή ώστε να παρακάμπτεται κάποιος κανονικός έλεγχος. Ένας τρόπος ώστε οι εταιρείες να εμποδίζουν τη δημιουργία των καταπακτών αυτών είναι να εφαρμόζουν ως συνήθη πρακτική τις επιθεωρήσεις κώδικα (code reviews).

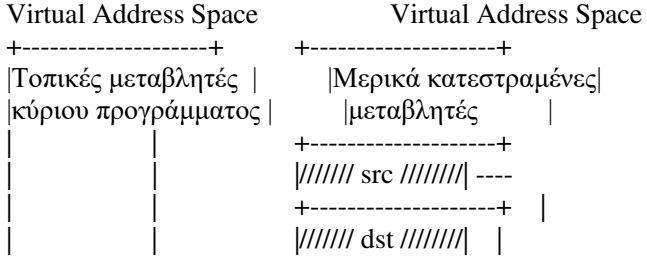
3) Παραπλανητική σύνδεση (login spoofing)

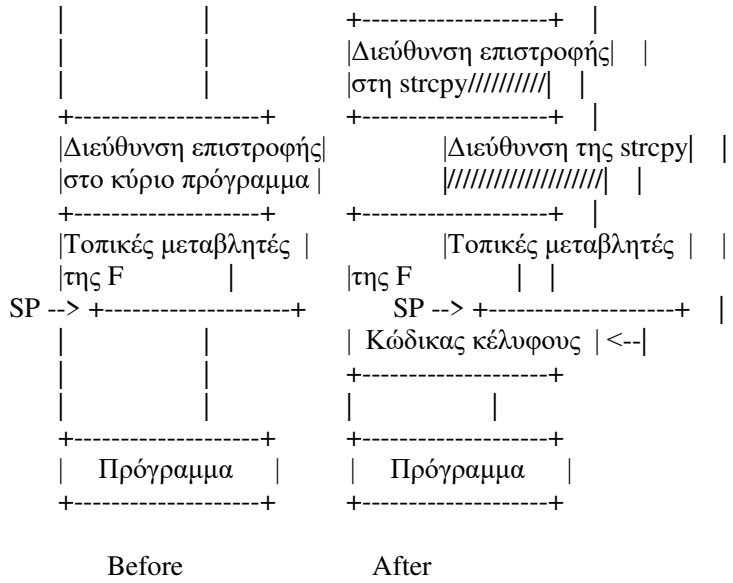
Κανονικά, όταν δεν είναι κανείς συνδεδεμένος σε κάποιο τερματικό UNIX ή ένα σταθμό εργασίας σε τοπικό δίκτυο, εμφανίζεται στην οθόνη ένα μήνυμα σύνδεσης. Όταν ένας χρήστης πληκτρολογήσει το όνομά του, το σύστημα ζητάει τον κωδικό πρόσβασης. Αν είναι σωστός, ο χρήστης συνδέεται και ξεκινάει κάποιο κέλυφος.
Ο κακόβουλος χρήστης γράφει ένα πρόγραμμα που εμφανίζει μία οθόνη ίδια με την προηγούμενη, εκτός από το ότι δεν εκτελείται το πρόγραμμα σύνδεσης στο σύστημα, αλλά ένα άλλο πρόγραμμα που έχει γραφεί από τον κακόβουλο χρήστη. Όταν κάποιος χρήστης έλθει και πληκτρολογήσει το όνομά του, το πρόγραμμα αποκρίνεται ζητώντας τον κωδικό πρόσβασης και απενεργοποιώντας την αντήχηση στην οθόνη. Μετά τη λήψη του ονόματος χρήστη και του κωδικού πρόσβασης, αυτά γράφονται σε κάποιο αρχείο και το ψεύτικο πρόγραμμα σύνδεσης στέλνει ένα σήμα για να τερματίσει το κέλυφός του.
Ο μόνος τρόπος για να αμυνθεί κανείς σε αυτό το σενάριο είναι να ξεκινάει η ακολουθία σύνδεσης με ένα συνδυασμό πλήκτρων που τα προγράμματα χρήστη δεν μπορούν να συλλάβουν (CTRL+ALT+DEL στα Windows).

93) Ποιοί οι τρόποι αξιοποίησης σφαλμάτων κώδικα για παράκαμψη ασφάλειας;

1) Υπερχείλιση προσωρινής μνήμης (buffer overflow)
2) Επιθέσεις με συμβολοσειρές μορφοποίησης (format string attack)
3) Επιθέσεις με επιστροφή στη libc

Σχεδόν όλα τα προγράμματα της C συνδέονται με τη (συνήθως κοινόχρηστη) βιβλιοθήκη libc, η οποία περιέχει τις κυριότερες συναρτήσεις τι οποίες χρειάζονται τα περισσότερα προγράμματα της C. Μία από τις συναρτήσεις αυτές είναι η strcpy, που αντιγράφει μία οποιαδήποτε συμβολοσειρά byte από κάποια διεύθυνση σε οποιαδήποτε άλλη διεύθυνση. Στόχος αυτής της επίθεσης είναι να ξεγελαστεί η strcpy και να αντιγράψει το πρόγραμμα του επιτιθέμενου, το οποίο συχνά ονομάζεται κώδικας κελύφους (shellcode), στο τμήμα δεδομένων (data segment), ώστε να εκτελεστεί από εκείνη τη θέση.
Αυτό που πρέπει να κάνει επίσης η επίθεση, εκτός από το να τοποθετήσει τον κώδικα κελύφους στη στοίβα, είναι να αντικαταστήσει τις τέσσερις σκιασμένες λέξεις:





Η χαμηλότερη από αυτές ήταν προηγουμένως η διεύθυνση επιστροφής στο κύριο πρόγραμμα, αλλά τώρα είναι η διεύθυνση της strcpy οπότε, όταν η f επιστρέψει, "επιστρέφει" στη strcpy. Στο σημείο αυτό, ο δείκτης στοίβας δείχνει σε μία κάλπικη διεύθυνση επιστροφής την οποία θα χρησιμοποιήσει η ίδια η strcpy όταν τελειώσει. Η διεύθυνση αυτή είναι η θέση που θα τοποθετηθεί ο κώδικας κελύφους. Οι δύο λέξεις επάνω από αυτή είναι οι διευθύνσεις προέλευσης και προορισμού για την αντιγραφή. Όταν τελειώσει η strcpy, ο κώδικας κελύφους θα βρίσκεται στη θέση στο (εκτελέσιμο) τμήμα δεδομένων και η strcpy θα "επιστρέψει" εκεί. Ο κώδικας κελύφους μπορεί να δημιουργήσει ένα κέλυφος για τον επιτιθέμενο που θα χρησιμοποιήσει αργότερα, ή μπορεί να ξεκινήσει ένα σενάριο παρακολούθησης μιας θύρας IP και αναμονής για εισερχόμενες διαταγές.

4) Επιθέσεις υπερχείλισης ακεραίων
5) Επιθέσεις με παρεμβολή κώδικα (code injection)
6) Επιθέσεις με κλιμάκωση προνομίων (privilege escalation attack)

93) Ποιό κακόβουλο λογισμικό ονομάζεται Δούρειος ίππος (Trojan Horse);

Η προσέγγιση κατά την οποία ο κακόβουλος χρήστης γράφει κάποιο έξυπνα χρήσιμο βοηθητικό πρόγραμμα και ενσωματώνει σε αυτό το κακόβουλο λογισμικό. Οι χρήστες θα κατεβάσουν με τη θέλησή τους και θα εγκαταστήσουν την εφαρμογή. Ως "δώρο", θα πάρουν και το κακόβουλο λογισμικό.
Το ωραίο της επίθεσης είναι ότι ο συγγραφέας του δε χρειάζεται να κάνει τίποτε για να διεισδύσει στον υπολογιστή του θύματος. Το θύμα κάνει τα πάντα μόνον του.

94) Ποιό κακόβουλο λογισμικό ονομάζεται Ιός (Virus);

Ιός είναι ένα πρόγραμμα που μπορεί να αναπαραχθεί μόνο του, προσθέτοντας τον κώδικά του σε κάποιο άλλο πρόγραμμα, παρόμοια με τον τρόπο που αναπαράγονται οι βιολογικοί ιοί. Επίσης, ο ιός μπορεί να κάνει και άλλα πράγματα εκτός από το να φροντίζει για την αναπαραγωγή του.

95) Πώς λειτουργούν οι ιοί;

Ο συγγραφέας του ιού, αφού τον γράψει, τον προσθέτει σε ένα πρόγραμμα στον υπολογιστή του χρησιμοποιώντας ένα εργαλείο που ονομάζεται σταγονόμετρο (dropper). Στη συνέχεια, διανέμει το μολυσμένο πρόγραμμα. Αφού εγκατασταθεί στον υπολογιστή του θύματος, ο ιός "κοιμάται" μέχρι να εκτελεστεί το μολυσμένο πρόγραμμα. Όταν αυτό ξεκινήσει, ο ιός συνήθως αρχίζει να μολύνει και άλλα προγράμματα στον υπολογιστή και μετά εκτελεί το "ωφέλιμο" φορτίο του (payload). Σε πολλές περιπτώσει, το φορτίο αυτό μπορεί να μην κάνει τίποτα μέχρι κάποια συγκεκριμένη ημερομηνία, ώστε να εξασφαλιστεί ότι ο ιός θα έχει διαδοθεί ευρέως πριν ανακαλυφθεί.

96) Ποιά είναι τα είδη ιών που διακρίνονται με βάση το στοιχείο που μολύνεται;

1) Συνοδευτικοί ιοί (companion virus)

Δεν μολύνει πραγματικά ένα πρόγραμμα, αλλά εκτελείται όταν εκτελείται το πρόγραμμα.

Στο MS-DOS, όταν ο χρήστης θέλει να εκτελέσει ένα πρόγραμμα από τη γραμμή εντολών, έστω Firefox, το λειτουργικό ψάχνει πρώτα να βρει ένα πρόγραμμα που να ονομάζεται Firefox.com. Αν δε μπορεί να το βρει, ψάχνει να βρει ένα πρόγραμμα που ονομάζεται Firefox.exe.
Ο συγγραφέας του ιού θα μπορούσε να δώσει στον ιό του το όνομα ενός γνωστού προγράμματος με την κατάληξη .exe. Ο χρήστης όταν εκτελέσει θέλει να εκτελέσει το αντίστοιχο πρόγραμμα, εν αγνοία του θα εκτελέσει πρώτα τον ιό, και έπειτα ο ιός, αφού ο ιός ολοκληρώσει την εργασία του, προκαλεί την εκτέλεση και του πραγματικού προγράμματος και ο χρήστης δεν καταλαβαίνει τίποτα.

Μία κάπως σχετική επίθεση χρησιμοποιεί την επιφάνεια εργασίας (desktop) των Windows, η οποία περιέχει συντομεύσεις (συμβολικούς συνδέσμους) προς προγράμματα. Ο ιός μπορεί να αλλάξει τον προορισμό μιας συντόμευσης ώστε αυτή να δείχνει στον ιό. Όταν ο χρήστης διπλοπατάει σε ένα εικονίδιο, εκτελείται ο ιός. Όταν ολοκληρώσει την εργασία του, ο ιός απλώς εκτελεί και το αρχικό πρόγραμμα.

2) Ιοί εκτελέσιμων προγραμμάτων

Ο απλούστερος από αυτούς τους ιούς απλώς αντικαθιστά το εκτελέσιμο πρόγραμμα με τον εαυτό του. Οι ιοί αυτοί ονομάζονται ιοί αντικατάστασης (overwriting viruses). Το πρόβλημα ενός ιού αντικατάστασης είναι ότι εντοπίζεται πολύ εύκολα. Συνεπώς, οι περισσότεροι ιοί προσκολλώνται στα προγράμματα και εκτελούν το έργο τους, αλλά επιτρέπουν στα προγράμματα να λειτουργούν κανονικά και μετά από αυτό. Αυτοί οι ιοί ονομάζονται παρασιτικοί ιοί (parasitic viruses).

Οι παρασιτικοί ιοί μπορούν να προσαρτηθούν στην αρχή, στο τέλος, ή στο μέσο του εκτελέσιμου προγράμματος. Αν ο ιός προσαρτήσει τον εαυτό του στην αρχή του προγράμματος πρέπει πρώτα να αντιγράψει το πρόγραμμα στη RAM, να γράψει τον εαυτό του στην αρχή του αρχείου, και στη συνέχεια να αντιγράψει το πρόγραμμα από τη RAM μετά από τον εαυτό του. Δυστυχώς, το πρόγραμμα δε θα εκτελεστεί στη νέα εικονική διεύθυνσή του, οπότε είτε ο

ιός πρέπει να το επανατοποθετήσει καθώς αυτό μετακινείται, ή να το επιστρέψει στην εικονική διεύθυνση 0 μετά από το τέλος της δικής του εκτέλεσης.
Για να αποφύγουν τις πολύπλοκες μεθόδους που απαιτούνται όταν η ενσωμάτωση γίνεται στην αρχή των προγραμμάτων, οι περισσότεροι ιοί προσαρτώνται στο τέλος των εκτελέσιμων προγραμμάτων αλλάζοντας το πεδίο αρχικής διεύθυνσης στην κεφαλίδα ώστε να δείχνει στην αρχή του ιού. Ο ιός θα εκτελεστεί τώρα σε διαφορετική εικονική διεύθυνση ανάλογα με το μολυσμένο πρόγραμμα που εκτελείται, αλλά αυτό σημαίνει ότι ο συγγραφέας πρέπει να εξασφαλίσει ότι ο ιός του θα είναι ανεξάρτητος από τη θέση, χρησιμοποιώντας σχετικές αντί για απόλυτες διευθύνσεις. Αυτό δεν είναι δύσκολο για έναν έμπειρο προγραμματιστή.

3) Ιοί που παραμένουν στη μνήμη (memory-resident viruses)

Ένας τυπικός ιός που παραμένει στη μνήμη καταλαμβάνει ένα από τα διανύσματα παγίδευσης ή διακοπής, αντιγράφοντας τα περιεχόμενα σε μία πρόχειρη μεταβλητή και τοποθετώντας εκεί τη διεύθυνσή της, κατευθύνοντας έτσι την παγίδευση ή τη διακοπή στον εαυτό του. Η καλύτερη επιλογή είναι η παγίδευση κλήσης συστήματος. Με αυτόν το τρόπο, ο ιός εκτελείται (σε κατάσταση πυρήνα) με κάθε κλήση συστήματος. Όταν τελειώσει απλώς ενεργοποιεί την πραγματική κλήση συστήματος με άλμα στην αποθηκευμένη διεύθυνση παγίδας.

4) Ιοί τομέα εκκίνησης (boot sector viruses)

Οι ιοί αυτοί αντικαθιστούν την κύρια εγγραφή εκκίνησης (master boot record - MBR), με καταστροφικά αποτελέσματα.
Συνήθως, οι ιοί τομέα εκκίνησης αντιγράφουν πρώτα τον πραγματικό τομέα εκκίνησης σε ένα ασφαλές σημείο στο δίσκο ώστε να μπορούν να ξεκινήσουν το λειτουργικό σύστημα όταν ολοκληρώσουν την εργασία τους. Μία επιλογή είναι να χρησιμοποιηθεί οποιοσδήποτε ελεύθερος τομέας στο δίσκο και στη συνέχεια να ενημερωθεί η λίστα ελαττωματικών τομέων ώστε να σημανθεί η κρυψώνα του ιού ως κατεστραμμένος τομέας.
Όταν ξεκινάει ο υπολογιστής, ο ιός αντιγράφει τον εαυτό του στη RAM, είτε στην κορυφή της είτε ανάμεσα στα αχρησιμοποίητα διανύσματα διακοπών. Στο σημείο αυτό ο υπολογιστής λειτουργεί σε κατάσταση πυρήνα, η MMU είναι απενεργοποιημένη, δεν υπάρχει λειτουργικό σύστημα, και δεν εκτελείται κανένα πρόγραμμα προστασίας από τους ιούς. Όταν ολοκληρώσει το έργο του, ξεκινάει το λειτουργικό σύστημα και συνήθως παραμένει στη μνήμη ώστε να παρακολουθεί τα τεκταινόμενα.

Ένα πρόβλημα είναι πως θα ξαναπάρει τον έλεγχο αργότερα. Ο συνήθης τρόπος είναι να εκμεταλλευθεί συγκεκριμένες γνώσεις που αφορούν τη διαχείριση του διανύσματος διακοπών από το λειτουργικό σύστημα.

5) Ιοί οδηγών συσκευής (device driver virus)

Οι οδηγοί συσκευών είναι απλώς εκτελέσιμα προγράμματα που βρίσκονται στο δίσκο και φορτώνονται κατά την εκκίνηση. Αν κάποιος από αυτούς μολυνθεί από έναν ιό, ο ιός θα φορτώνεται νόμιμα κάθε φορά που γίνεται εκκίνηση του συστήματος. Ακόμη καλύτερα, οι οδηγοί εκτελούνται σε κατάσταση πυρήνα και, μετά από την φόρτωση ενός οδηγού, αυτός καλείται αμέσως

δίνοντας στον ιό τη δυνατότητα να καταλάβει το διάνυσμα παγιδεύσεων των κλήσεων συστήματος. Αυτό και μόνον αποτελεί συνήθως ένα ισχυρό επιχείρημα υπέρ της εκτέλεσης των οδηγών συσκευών ως προγραμμάτων σε κατάσταση χρήστη.

6) Μακροϊοί (macro viruses)

Πολλά προγράμματα, όπως τα Word και Excel, επιτρέπουν στους χρήστες να γράφουν μακροεντολές που ομαδοποιούν διάφορες διαταγές και τις εκτελούν κατόπιν με το πάτημα ενός πλήκτρου. Οι μακροεντολές μπορούν επίσης να τοποθετηθούν στα μενού ώστε, όταν επιλέγεται κάποιο από αυτά, να εκτελείται η μακροεντολή.
Ο κακόβουλος χρήστης γράφει ένα έγγραφο στο Word και δημιουργεί μια μακροεντολή την οποία προσαρτάει στην συνάρτηση OPEN FILE. Η μακροεντολή περιέχει ένα μακροϊό. Στη συνέχεια,, στέλνει το έγγραφο μέσω ηλεκτρονικού ταχυδρομείου στο θύμα, το οποίο φυσικά το ανοίγει. Το άνοιγμα του εγγράφου προσκαλεί την εκτέλεση της μακροεντολής OPEN FILE. Καθώς η μακροεντολή μπορεί να περιέχει οποιδήποτε πρόγραμμα, μπορεί να κάνει οτιδήποτε, όπως να μολύνει άλλα έγγραφα του Word, να διαγράψει αρχεία, και άλλα πολλά.

7) Ιοί πηγαίου κώδικα (source code viruses)

Οι πιο φορητοί ιοί είναι οι ιοί πηγαίου κώδικα. Αντί να ψάχνουν για δυαδικά εκτελέσιμα αρχεία, ψάχνουν για προγράμματα C. Αυτή η τροποποίηση, απαιτεί την αλλαγή μόνο μίας/δύο γραμμών. Απαιτείται η προσθήκη της γραμμής:

```
#include <virus.h>
```

και της γραμμής:

```
run_virus();
```

για να ενεργοποιηθεί ο ιός. Η απόφαση για το που θα τοποθετηθεί αυτή η γραμμή απαιτεί κάποιες ικανότητες συντακτικής ανάλυσης κώδικα C, επειδή πρέπει να βρίσκεται σε θέση όπου συντακτικά επιτρέπονται οι κλήσεις διαδικασιών αλλά όχι εκεί που ο κώδικας είναι "νεκρός" (π.χ., μετά από μία εντολή return).
Όταν εκτελεστεί το πρόγραμμα, θα κληθεί ο ιός. Ο ιός μπορεί να κάνει ό,τι θέλει, όπως για παράδειγμα να ψάξει για άλλα προγράμματα C ώστε να τα μολύνει. Αν βρει κάποιο, μπορεί απλώς να τοποθετήσει τις δύο γραμμές που είδαμε προηγουμένως, αλλά αυτό λειτουργεί μόνο στον τοπικό υπολογιστή όπου υποτίθεται ότι το virus.h έχει ήδη εγκατασταθεί.

97) Ποιοί είναι οι τρόποι εξάπλωσης των ιών;

1) Λογισμικό δοκιμαστικής χρήσης (shareware)

Ο συγγραφέας του ιού, το προσθέτει σε κάποιο πρόγραμμα που έχει και ξεκινάει τη διανομή του προγράμματος, τοποθετώντας το για παράδειγμα σε κάποια τοποθεσία Ιστού με λογισμικό δοκιμαστικής χρήσης (shareware). Στο τέλος κάποιος θα κατεβάσει το πρόγραμμα και θα το εκτελέσει.

2) Τοπικό δίκτυο

Ο κακόβουλος χρήστης μπορεί να γράψει έναν ιό που θα ελέγχει αν ο μολυσμένος υπολογιστής είναι συνδεδεμένος σε τοπικό δίκτυο, κάτι πολύ πιθανό αν ανήκει σε κάποια εταιρεία ή πανεπιστήμιο. Ο ιός μπορεί στη συνέχεια να ξεκινήσει τη μόλυνση απροστάτευτων αρχείων στους διακομιστές του τοπικού δικτύου.

3) Ηλεκτρονική ταχυδρόμηση μολυσμένου προγράμματος ως συνημμένο

Ακόμη και όσοι δε θέλουν να εκτελέσουν ένα πρόγραμμα που τους έστειλε ένας ξένος μπορεί να μη γνωρίζουν ότι, αν ανοίξουν το συνημμένο, θα απελευθερώσουν έναν ιό στον υπολογιστή τους. Ακόμα χειρότερα, ο ιός μπορεί να ψάξει για το βιβλίο διευθύνσεων (address book) του χρήστη και να ταχυδρομήσει το εαυτό του σε οποιονδήποτε χρήστη υπάρχει εκεί, συνήθως με μία γραμμή θέματος που μια νόμιμη ή ενδιαφέρουσα.

98) Ποιό κακόβουλο λογισμικό ονομάζεται Σκουλήκι (Worm);

Είναι ένα αυτοαναπαραγόμενο και κακόβουλο πρόγραμμα υπολογιστή, το οποίο χρησιμοποιεί δίκτυο υπολογιστών για να στείλει αντίγραφα του εαυτού του σε άλλους κόμβους (υπολογιστές του δικτύου) και μπορεί να το πράξει χωρίς την παρέμβαση του χρήστη. Το γεγονός αυτό οφείλεται σε κενά ασφαλείας του υπολογιστή προορισμού

99) Ποιό κακόβουλο λογισμικό ονομάζεται Λογισμικό Κατασκοπίας (Spyware);

Σε γενικές γραμμές, το λογισμικό κατασκοπίας είναι λογισμικό το οποίο φορτώνεται κρυφά σε ένα PC χωρίς να το ξέρει ο ιδιοκτήτης του και εκτελείται στο παρασκήνιο κάνοντας διάφορα πράγματα πίσω από την πλάτη του τελευταίου.

Έχει 4 χαρακτηριστικά:

1) Κρύβεται, ώστε να μην μπορεί το θύμα να το εντοπίσει έυκολα
2) Συγκεντρώνει στοιχεία σχετικά με το χρήστη
3) Μεταδίδει τις πληροφορίες που έχει συγκεντρώσει σε ένα μακρινό κύριό του
4) Προσπαθεί να επιβιώσει από αποφασιστικές προσπάθειες αφαίρεσής του

Έχει 3 γενικές κατηγορίες:

1) Μάρκετινγκ

Απλώς συλλέγει πληροφορίες και τις στέλνει στον κύριό του, συνήθως με σκοπό την καλύτερη στόχευση της διαφήμισης σε συγκεκριμένες μηχανές.

2) Παρακολούθηση

Οι εταιρίες τοποθετούν εσκεμμένα λογισμικό κατασκοπίας στις μηχανές των υπαλλήλων τους ώστε να παρακολουθούν τι κάνουν και ποιες τοποθεσίες Ιστού επισκέπτονται.

3) Κακόβουλο λογισμικό

Η μολυσμένη μηχανή γίνεται τμήμα μιας στρατιάς ζόμπι η οποία περιμένει τη διαταγή του κυρίου της για να εφορμήσει.

100) Ποιοί οι τρόποι εξάπλωσης του λογισμικού κατασκοπίας;

1) Μέσω ενός Δούρειου Ίππου
2) Μολυσμένη γραμμή εργαλείων (brower plug-ins/addons)
3) Μέσω των στοιχείων ελέγχου activeX με στόχο τον Internet Explorer

101) Τι κάνει συνήθως το λογισμικό κατασκοπίας;

1. Αλλαγή της αρχικής σελίδας του φυλλομετρητή
2. Τροποποίηση της λίστας αγαπημένων (σελιδοδεικτών) του φυλλομετρητή
3. Προσθήκη νέων γραμμών εργαλείων στο φυλλομετρητή
4. Αλλαγή του προεπιλεγμένου προγράμματος αναπαραγωγής πολυμέσων του χρήστη
5. Αλλαγή της προεπιλεγμένης μηχανής αναζήτησης του χρήστη
6. Προσθήκη νέων εικονιδίων στην επιφάνειας εργασίας των Windows
7. Αντικατάσταση διαφημίσεων σε ιστοσελίδες με αυτές που επιλέγει λογισμικό κατασκοπίας
8. Προσθήκη διαφημίσεων σε ιστοσελίδες με αυτές που επιλέγει το λογισμικό κατασκοπίας
9. Δημιουργία ενός συνεχούς και αδιάλειπτου ρεύματος αναδυόμενων διαφημίσεων

102) Ποιό κακόβουλο λογισμικό ονομάζεται Rootkit;

Rootkit είναι ένα πρόγραμμα ή σύνολο προγραμμάτων και αρχείων που προσπαθεί να κρύψει την ύπαρξή του, ακόμη και μετά από τις αποφασιστικές προσπάθειες του ιδιοκτήτη της μολυσμένης μηχανής να το εντοπίσει και να το αφαιρέσει. Συνήθως, το rootkit περιέχει κάποιο κακόβουλο λογισμικό το οποίο επίσης κρύβεται. Τα rootkit μπορούν να εγκατασταθούν με οποιαδήποτε από τις μεθόδους που περιγράψαμε μέχρι τώρα, όπως από ιούς, σκουλήκια, και λογισμικό κατασκοπίας καθώς και με άλλους τρόπους.

103) Ποιοί οι τύποι των Rootkits και ποιοί οι τρόποι εντοπισμού τους;

1. Rootkit υλικολογισμικού (firmware rootkits)

   Θεωρητικά τουλάχιστον, ένα rootkit μπορεί να κρυφτεί γράφοντας στο BIOS ένα αντίγραφό του. Ένα τέτοιο rootkit μπορεί να πάρει τον έλεγχο κατά την εκκίνηση της μηχανής και όποτε καλείται μία συνάρτηση του BIOS. Αυτός ο τύπος δεν έχει παρατηρηθεί ακόμα στη πράξη.

2. Rootkit υπερεπόπτη (hypervisor rootkits)

   Ένα εξαιρετικά ύπουλο είδος rootkit, που μπορεί να εκτελέσει το λειτουργικό σύστημα και όλες τις εφαρμογές του σε μία εικονική μηχανή υπό τον έλεγχό του. Αυτό το είδος rootkit συνήθως τροποποιεί τη σειρά εκκίνησης (boot sequence) ώστε, όταν ξεκινάει η μηχανή, να εκτελείται ο υπερεπόπτης απευθείας στο υλικό και στη συνέχεια να ξεκινάει το λειτουργικό σύστημα και τις εφαρμογές του σε μία εικονική μηχανή.
   Το πλεονέκτημα της μεθόδου αυτής, όπως και της προηγούμενης, είναι ότι δεν κρύβεται τίποτε στο λειτουργικό σύστημα, τις βιβλιοθήκες, ή τα προγράμματα, οπότε οι ανιχνευτές rootkit που ψάχνουν στις θέσεις αυτές

δε βρίσκουν τίποτε.

- Εντοπισμός:

Βασίζεται στην παρατήρηση πως και ο ίδιος ο υπερεπόπτης χρησιμοποιεί φυσικούς πόρους, και η απώλεια των πόρων αυτών μπορεί να εντοπιστεί. Για παράδειγμα, ο ίδιος ο υπερεπόπτης χρειάζεται να χρησιμοποιήσει κάποιες καταχωρίσεις της TLB, ανταγωνιζόμενος τη φυσική μηχανή γι αυτούς τους σπάνιους πόρους. Ένα πρόγραμμα ανίχνευσης θα μπορούσε να πιέσει την TLB, να παρατηρήσει την απόδοση, και να τη συγκρίνει με προηγούμενες μετρήσεις της απευθείας στο υλικό.

Ένας άλλος τρόπος θα μπορούσε να είναι να εξεταστεί ο χρόνος που απαιτείται για την εκτέλεση προνομιούχων εντολών, ειδικά αυτών που απαιτούν μόνο λίγους κύκλους ρολογιού στο πραγματικό υλικό και εκατοντάδες κύκλους ρολογιού όταν πρέπει να εξομοιωθούν.

3. Rootkit πυρήνα (kernel rootkits)

Το πιο κοινό είδος rootkit προς το παρόν είναι αυτό που προσβάλλει το λειτουργικό σύστημα και κρύβεται σε αυτό ως οδηγός συσκευής ή υπομονάδα πυρήνα με δυνατότητα φόρτωσης. Το rootkit μπορεί εύκολα να αντικαταστήσει ένα μεγάλο, σύνθετο, και συχνά τροποποιούμενο οδηγό συσκευής με ένα νέο που περιέχει τον παλιό μαζί με το rootkit.

- Εντοπισμός:

Ένας τρόπος είναι να εκκινηθεί ο υπολογιστής από κάποιο έμπιστο εξωτερικό μέσω όπως το πρωτότυπο CD-ROM/DVD ή μία μνήμη USB. Στη συνέχεια μπορεί να σαρωθεί ο δίσκος με τη βοήθεια ενός προγράμματος αντιμετώπισης των rootkit, χωρίς το φόβο παρέμβασης του ίδιου του rootkit στη σάρωση.

4. Rootkit βιβλιοθήκης (library rootkits)

Άλλη μία θέση όπου μπορεί να κρυφτεί ένα rootkit είναι η βιβλιοθήκη του συστήματος, όπως για παράδειγμα η libc στο Linux. Η θέση αυτή δίνει στο κακόβουλο λογισμικό τη δυνατότητα να εξετάζει τα ορίσματα και να τις επιστρεφόμενες τιμές των κλήσεων συστήματος, και να τις τροποποιεί κατάλληλα ώστε να παραμένει κρυμμένο.

5. Rootkit εφαρμογών (application rootkits)

Ακόμα μία κρυψώνα rootkit είναι μέσα σε ένα μεγάλο πρόγραμμα εφαρμογής, και ιδιαίτερα κάποιο που δημιουργεί πολλά αρχεία όταν εκτελείται (προφίλ χρήστη, αρχεία προεπισκόπησης εικόνων, κλπ.). Αυτά τα νέα αρχεία είναι πολύ καλές θέσεις για να κρυφτεί κάτι, ενώ κανένας δε θεωρεί παράξενη την ύπαρξή τους.

- Εντοπισμός των rootkits βιβλιοθηκών και εφαρμογών:

Αν το λειτουργικό σύστημα έχει φορτωθεί από ένα εξωτερικό μέσο και μπορείτε να το εμπιστευθείτε, οι κατακερματισμοί τους μπορούν να

συγκριθούν με κατακερματισμούς που είναι γνωστοί ως σωστοί και είναι αποθηκευμένοι σε ένα CD-ROM.

104) Αναφέρετε τρόπους άμυνας από κακόβουλο λογισμικό.

1) Τείχη προστασίας

Τα τείχη προστασίας υπάρχουν σε δύο βασικές ποικιλίες: υλικού και λογισμικού. Οι εταιρείες που έχουν να προστατεύσουν τοπικά δίκτυα συνήθως προτιμούν τα τείχη προστασίας υλικού, οι ιδιώτες χρήστες στο σπίτι τους επιλέγουν τα τείχη προστασίας λογισμικού.

Στη πράξη, τα τείχη προστασίας συνδυάζονται συχνά με δρομολογητές (routers), διατάξεις μετάφρασης διευθύνσεων δικτύου, συστήματα ανίχνευσης εισβολέων, και άλλα πράγματα, αλλά εδώ θα εστιάσουμε την ανάλυσή μας στις λειτουργίες του τείχους προστασίας.

Τα τείχη προστασίας ρυθμίζονται με βάση κανόνες που περιγράφουν τι επιτρέπεται και τι δεν επιτρέπεται. Ο ιδιοκτήτης του τείχους προστασίας μπορεί να αλλάξει τους κανόνες αυτούς, συνήθως μέσω μιας διασύνδεσης Ιστού.

1) Μη καταστασιακό τείχος προστασίας (stateless firewall)

Επιθεωρείται η κεφαλίδα κάθε διερχόμενου πακέτου και λαμβάνεται η απόφαση αν θα επιτραπεί η θα απαγορευτεί η διέλευση του πακέτου, με βάση μόνο τις πληροφορίες της κεφαλίδας του και τους κανόνες του τείχους προστασίας. Οι πληροφορίες στην κεφαλίδα του πακέτου περιλαμβάνουν τις διευθύνσεις IP προέλευσης και προορισμού, τις θύρες προέλευσης και προορισμού, τον τύπο της υπηρεσίας και το πρωτόκολλο.

2) Καταστασιακό τείχος προστασίας (statefull firewall)

Παρακολουθούν και καταγράφουν τις συνδέσεις και την κατάστασής τους. Αυτά τα τείχη προστασίας είναι καλύτερα στην αντιμετώπιση κάποιων τύπων επιθέσεων, και ιδιαίτερα αυτών που έχουν σχέση με την εγκαθίδρυση συνδέσεων.

2) Τεχνικές εναντίον των ιών και προστασίας των ιών από αυτές

Τα τείχη προστασίας προσπαθούν να κρατήσουν το κακόβουλο λογισμικό έξω από τον υπολογιστή. Αλλά μπορεί να μην τα καταφέρουν για πολλούς λόγους. Στην περίπτωση αυτή, η επόμενη γραμμή άμυνας αποτελείται από τα προγράμματα αντιμετώπισης του κακόβουλου λογισμικού, που συχνά έχουν το όνομα αντιβιοτικά προγράμματα (antivirus programs).

1) Σαρωτές ιών (virus scanners)

Οι εταιρείες αντιβιοτικού λογισμικού διαθέτουν εργαστήρια στα οποία αφοσιωμένοι επιστήμονες εργάζονται ευσυνείδητα για να εντοπίζουν και να κατανοήσουν τους νέους ιούς. Το πρώτο βήμα είναι να μολύνουν με τον ιό ένα πρόγραμμα που δεν κάνει τίποτε, το οποίο συχνά λέγεται

αρχείο-τράγος (goat file), ώστε να αποκτήσουν ένα αντίγραφο του ιού στην πιο καθαρή του μορφή. Το επόμενο βήμα είναι να κατασκευάσουν μία ακριβή λίστα του κώδικα του ιού και να την καταχωρίσουν στη βάση δεδομένων με τους γνωστούς ιούς.
Αφού ένα αντιβιοτικό πρόγραμμα εγκατασταθεί στον υπολογιστή ενός πελάτη, το πρώτο πράγμα που κάνει είναι να σαρώσει κάθε εκτελέσιμο αρχείο στο δίσκο, ψάχνοντας για οποιονδήποτε ιό έχει στη βάση δεδομένων του.

2) Ελεγκτές ακεραιότητας (integrity checkers)

Ένα αντιβιοτικό πρόγραμμα που λειτουργεί με αυτόν τον τρόπο ελέγχει πρώτα το σκληρό δίσκο για ιούς. Όταν πειστεί ότι ο δίσκος είναι καθαρός, υπολογίζει ένα άθροισμα ελέγχου (checksum) για κάθε εκτελέσιμο αρχείο. Ο αλγόριθμος του αθροίσματος ελέγχου μπορεί να είναι κάτι τόσο απλό όσο ο χειρισμός όλων των λέξεων του κώδικα του προγράμματος ως ακεραίων των 32 ή των 64 bit και την πρόσθεσή τους, αλλά μπορεί επίσης να είναι ένας σχεδόν αδύνατον να αντιστραφεί κρυπτογραφικός κατακερματισμός. Στη συνέχεια γράφει τη λίστα των αθροισμάτων ελέγχου, για όλα τα σχετικά αρχεία ενός καταλόγου, σε ένα αρχείο με το όνομα checksum το οποίο τοποθετεί στον ίδιο κατάλογο. Την επόμενη φορά που εκτελείται, υπολογίζει ξανά όλα τα αθροίσματα ελέγχου και ελέγχει αν ταιριάζουν με τα περιεχόμενα του αρχείου checksum.

3) Ελεγκτές συμπεριφοράς (behavioral checkers)

Με αυτή τη μέθοδο, το αντιβιοτικό πρόγραμμα παραμένει στη μνήμη κατά τη λειτουργία του υπολογιστή και αναχαιτίζει όλες τις κλήσεις συστήματος. Η ιδέα είναι ότι μπορεί πλέον να παρακολουθήσει όλες τις δραστηριότητες και να προσπαθήσει να συλλάβει οτιδήποτε μοιάζει ύποπτο. Για παράδειγμα, κανένα κανονικό πρόγραμμα δεν θα προσπαθούσε να αντικαταστήσει τον τομέα εκκίνησης, οπότε μια τέτοια προσπάθεια αποτελεί οπωσδήποτε έργο ιού.

4) Αποφυγή ιών

1) Επιλογή ασφαλούς λειτουργικού συστήματος
2) Εγκατάσταση πακέτων λογισμικού μόνον από αξιόπιστους κατασκευαστές
3) Αγορά ενός καλού πακέτου αντιβιοτικού λογισμικού
4) Αποφυγή ανοίγματος αρχείων του ηλεκτρονικού ταχυδρομείου
5) Συχνά αντίγραφα ασφαλείας των κρίσιμων αρχείων σε κάποιο εξωτερικό μέσο
6) Αποφυγή της εκτέλεσης οποιουδήποτε νέου, δωρεάν λογισμικού από άγνωστες πηγές.

3) Υπογραφή κώδικα

Μία τελείως διαφορετική προσέγγιση για την αποφυγή του κακόβουλου λογισμικού είναι η εκτέλεση μόνο μη τροποποιημένου λογισμικού από αξιόπιστους πωλητές λογισμικού.
Ένας τρόπος για να γνωρίζει ο χρήστης σε ποιόν ανήκει το λογισμικό που έχει στη κατοχή του, και που γνωρίζει ευρεία χρήση είναι η ψηφιακή υπογραφή. Αν ο χρήστης εκτελεί προγράμματα, συνδεόμενες υπομονάδες, οδηγούς συσκευών, στοιχεία ελέγχου activeX, και άλλα είδη λογισμικού

γραμμένα και υπογεγραμμένα μόνον από έμπιστες πηγές, οι πιθανότητες να αντιμετωπίσει προβλήματα είναι πολύ λιγότερες.
Η υπογραφή κώδικα βασίζεται στην κρυπτογραφία δημόσιου κλειδιού. Ο κατασκευαστής λογισμικού παράγει ένα ζεύγος (δημόσιο κλειδί, ιδιωτικό κλειδί), και δημοσιεύει το πρώτο ενώ φυλάει σαν τα μάτια του το δεύτερο. Για να υπογράψει ένα προϊόν λογισμικού, ο κατασκευαστής υπολογίζει πρώτα μία συνάρτηση κατακερματισμού του κώδικα για να πάρει έναν 128 bit 160 bit, ή 256 bit, ανάλογα με το αν χρησιμοποιείται MD5, SHA-1, ή SHA-256. Στη συνέχεια υπογράφει την τιμή κατακερματισμού κρυπτογραφώντας τη με το ιδιωτικό κλειδί (στην πραγματικότητα, την αποκρυπτογραφεί). Αυτή η υπογραφή συνοδεύει το λογισμικό όπου και να πάει.
Όταν ο χρήστης πάρει το λογισμικό, εφαρμόζεται σε αυτό η συνάρτηση κατακερματισμού και το αποτέλεσμα αποθηκεύεται. Στη συνέχεια αποκρυπτογραφείται η συνοδευτική υπογραφή με το δημόσιο κλειδί του κατασκευαστή και συγκρίνεται η τιμή της συνάρτησης κατακερματισμού που δίνει ο κατασκευαστής με την τιμή που υπολογίστηκε.

4) Φυλάκιση (jailing)

Το νέο πρόγραμμα εκτελείται ως μία διεργασία. Μία έμπιστη διεργασία (συστήματος) παρακολουθεί τη συμπεριφορά της νέας. Όταν η νέα διεργασία κάνει μία κλήση συστήματος, αντί να εκτελεστεί η κλήση συστήματος, ο έλεγχος μεταφέρεται στη δεύτερη διεργασία (μέσω παγίδευσης πυρήνα) και μεταβιβάζονται σε αυτή ο αριθμός της κλήσης συστήματος και οι παράμετροί της.

5) Ανίχνευση εισβολής με βάση μοντέλο (model-based intrusion detection)

Η διεργασία "δεσμοφύλακας" μπορεί να δημιουργήσει ένα π.χ. στατικό γράφο με τις κλήσεις συστήματος του κώδικα του προγράμματος. Εάν ο επιτιθέμενος προκαλέσει κάποια κλήση συστήματος, π.χ. χρησιμοποιώντας υπερχείλιση προσωρινής μνήμης, που δε συμφωνεί με το γράφο, η διεργασία "δεσμοφύλακας" μπορεί να "σκοτώσει" τη διεργασία αυτή.

6) Ενθυλάκωση κινητού κώδικα

Στις μέρες μας όλο και περισσότερες ιστοσελίδες περιέχουν μικρά προγράμματα που ονομάζονται μικροεφαρμογές (applets). Όταν ο χρήστης κατεβάζει μία ιστοσελίδα που περιέχει μικροεφαρμογές, οι τελευταίες μεταφέρονται στον υπολογιστή του και εκτελούνται. Αυτές οι μικροεφαρμογές είναι ένα παράδειγμα κινητού κώδικα (mobile code).

Τρόποι χειρισμού μικροεφαμογών:

1) Αμμοπαγίδα (sandboxing)

Προσπαθεί να περιορίσει κάθε μικροεφαρμογή σε ένα συγκεκριμένο εύρος εικονικών διευθύνσεων, οι οποίες της παραχωρούνται κατά το χρόνο εκτέλεσης. Διαιρεί το χώρο εικονικών διευθύνσεων σε περιοχές ίσου μεγέθους, τις οποίες θα αποκαλούμε αμμοπαγίδες. Κάθε αμμοπαγίδα πρέπει να έχει την ιδιότητα όλες οι διευθύνσεις της να μοιράζονται κάποια συμβολοσειρά bit υψηλής τάξης.

Η βασική ιδέα είναι να εξασφαλιστεί ότι μία μικροεφαρμογή δε θα μπορεί να διακλαδωθεί σε κώδικα έξω από την αμμοπαγίδα του κώδικά της ή να προσπελάσει δεδομένα έξω από την αμμοπαγίδα των δεδομένων της.

2) Διερμηνεία (interpretation)

Τεχνική κατά την οποία δεν επιτρέπεται στις μικροεφαρμογές να αποκτήσουν πραγματικό έλεγχο του υλικού. Όταν κατεβαίνουν στον υπολογιστή του χρήστη, εισάγονται σε ένα διερμηνευτή JVM στο εσωτερικό του φυλλομετρητή. Ο διερμηνευτής ελέγχει κάθε εντολή πριν την εκτελέσει. Το γεγονός αυτό του δίνει τη δυνατότητα να ελέγξει αν η διεύθυνση είναι έγκυρη. Ακόμα, συλλαμβάνονται και διερμηνεύονται οι κλήσεις συστήματος. Ο τρόπος χειρισμού αυτών των κλήσεων είναι ζήτημα που εξαρτάται από την πολιτική ασφαλείας.

105) Γιατί η σχεδίαση λειτουργικών συστημάτων είναι δύσκολη;

1) Τα λειτουργικά συστήματα έχουν γίνει εξαιρετικά ογκώδη
2) Πρέπει να χειρίζονται τον ταυτοχρονισμό (concurrency)
3) Πρέπει να αντιμετωπίσουν πιθανόν εχθρικούς χρήστες
4) Πρέπει να παρέχει εμπιστευτικότητα μεταξύ χρηστών
5) Πρέπει να έχουν μεγάλη διάρκεια ζωής
6) Οι σχεδιαστές δεν έχουν ένα πλήρες πλάνο της χρήσης των λειτουργικών συστημάτων που δημιουργούν
7) Σχεδιάζονται με σκοπό να είναι φορητά, να εκτελούνται επάνω σε πολλές πλατφόρμες υλικού
8) Έχουν την ανάγκη για αναδρομική συμβατότητα (backward compatibility)

106) Ποιές είναι η καθοδηγητικές αρχές σχεδίασης διασυνδέσεων;

1. Απλότητα
2. Πληρότητα
3. Αποδοτικότητα

107) Σχολιάστε περί Μηχανισμού και Πολιτικής.

Μία αρχή που βοηθάει τη συνοχή της αρχιτεκτονικής, λέει ότι ο μηχανισμός πρέπει να διαχωρίζεται από την πολιτική.

Η πολιτική απαντάει στο "Τι θα γίνει, πότε, ποιός θα το κάνει, κτλ", ενώ ο μηχανισμός απαντάει στο "Πως θα γίνει".

Ένα παράδειγμα:
Σχετικά με τη δυνατότητα φόρτωσης υπομονάδων στον πυρήνα. Ο μηχανισμός αφορά τον τρόπο εισαγωγής και σύνδεσής του, τις κλήσεις που μπορούν να κάνουν, και τις κλήσεις που μπορούν να εφαρμοστούν σε αυτές. Η πολιτική είναι το ποιος επιτρέπεται να φορτώνει ποιές υπομονάδες στον πυρήνα.

Ένα ακόμα παράδειγμα:
Σχετικά με τη σελιδοποίηση. Ο μηχανισμός περιλαμβάνει τη διαχείριση της MMU, τη διατήρηση λιστών με κατειλημμένες και ελεύθερες σελίδες, και

κώδικα για τη μετακίνηση σελίδων από και προς το δίσκο. Η πολιτική αποφασίζει τι πρέπει να γίνει όταν προκύψει κάποιο σφάλμα σελίδας, τον αλγόριθμο επιλογής σελίδας, κτλ.

108) Σχολιάστε περί Ορθογωνικότητας.

Η δυνατότητα να συνδυάζονται ανεξάρτητα ξεχωριστές έννοιες ονομάζεται ορθογωνικότητα (orthogonality)

Ένα παράδειγμα:
Στο UNIX η δημιουργία διεργασίας γίνεται εδώ σε δύο βήματα: πρώτα καλείται η fork και μετά η exec. Η δημιουργία του νέου χώρου διευθύνσεων και η φόρτωση μιας νέας εικόνας μνήμης σε αυτό είναι ξεχωριστές ενέργειες, που επιτρέπουν την εκτέλεση άλλων ενεργειών στο μεταξύ (όπως ο χειρισμός περιγραφέων αρχείων).

Ως γενικό κανόνα θα λέγαμε ότι, έχοντας λίγα ορθογωνικά στοιχεία που μπορούν να συνδυαστούν με πολλούς τρόπους, οδηγούμαστε σε μικρά, απλά, και κομψά συστήματα.