Ασκήσεις -Ερωτήσεις
1ου Κεφαλαίου

Άσκηση 1.1 (Πρ. 1-2)

(α) Τι είναι ο πολυπρογραμματισμός (multiprogramming);

(β) Τι είναι η παροχέτευση εισόδου και εξόδου (input spooling / output spooling);

Απάντηση 1.1 (Πρ. 1-2)

(α) Τι είναι ο πολυπρογραμματισμός (multiprogramming); Σε τι βοηθάει;

- Πολυπρογραμματισμός (Multiprogramming) είναι η λειτουργία της ταχύτατης μετάβασης της CPU, μεταξύ πολλών διεργασιών που βρίσκονται στη μνήμη για εκτέλεση. Η χρήση του πολυπρογραμματισμού επιτρέπει στη CPU να έχει μεγαλύτερο βαθμό χρησιμοποίησης (utilization) και να μην βρίσκεται σε αδράνεια, κατά τη φάση που οι διεργασίες αναμένουν για Ε/Ε.

(β) Τι είναι η παροχέτευση εισόδου και εξόδου (input spooling / output spooling);

- Η παροχέτευση εισόδου είναι η λειτουργία της ανάγνωσης εργασιών (jobs) έτσι ώστε να επιτρέπεται η σειριακή εκτέλεσή τους και να μειωθεί ο χρόνος αναμονής της CPU. Η παροχέτευση εξόδου είναι η λειτουργία η οποία επιτρέπει τη δημιουργία μίας σειράς εργασιών εξόδου (π.χ. αποθήκευση αρχείων σε κάποια ουρά αναμονής πριν την εκτύπωσή τους) . Η 1η δεν έχει πλέον εφαρμογή, η 2η έχει.

Άσκηση 1.2 (Πρ. 5)

Ο βασικός λόγος που άργησαν τα GUI να υιοθετηθούν είναι το κόστος του υλικού που χρειαζόταν για την υποστήριξή τους.

(α) Πόση μνήμη RAM χρειάζεται για να υποστηρίξει μια μονόχρωμη οθόνη κειμένου με χωρητικότητα χαρακτήρων 25 γραμμές × 80 στήλες;

(β) Πόση RAM χρειάζεται για μία έγχρωμη οθόνη bitmap με ανάλυση 1024 × 768 pixel των 24 bit; Πόσο το κόστος σε μνήμη σε τιμές 1980 (5 $/KB μνήμης); Πόσο είναι το κόστος σήμερα (με μονάδα κόστους 10^{-5} $ / KB μνήμης);

Απάντηση 1.2 (Πρ. 8)

(α) 25×80 = 2000 χαρακτήρες. Αν κάθε χαρακτήρας είναι 1byte, τότε οι απαιτήσεις μνήμης RAM για την υποστήριξη της προβολής της οθόνης είναι περίπου 2KB μνήμης.

(β) Η οθόνη αποτελείται από 1024×768 = 786.432 pixel.

Κάθε pixel είναι 24 bit άρα 24/8 = 3 byte.

Συνεπώς 786.432×3 = 2.359.296 byte ή 2.304 KB.

Η απαιτούμενη μνήμη RAM σε τιμές 1980 είναι:

2.304 KB × 5 $/KB = 11.520$

Σε σημερινές τιμές: 2.304 KB × 10^{-5} $/KB = 0,02304$

Άσκηση 1.3 (Πρ. 7)

Ποιες από τις παρακάτω εντολές μπορούν να επιτραπούν μόνο σε κατάσταση λειτουργίας πυρήνα (kernel mode);

(α) Απενεργοποίηση όλων των διακοπών (interrupts)

(β) Ανάγνωση ώρας από το ρολόι συστήματος

(γ) Απόδοση τιμής στο ρολόι συστήματος

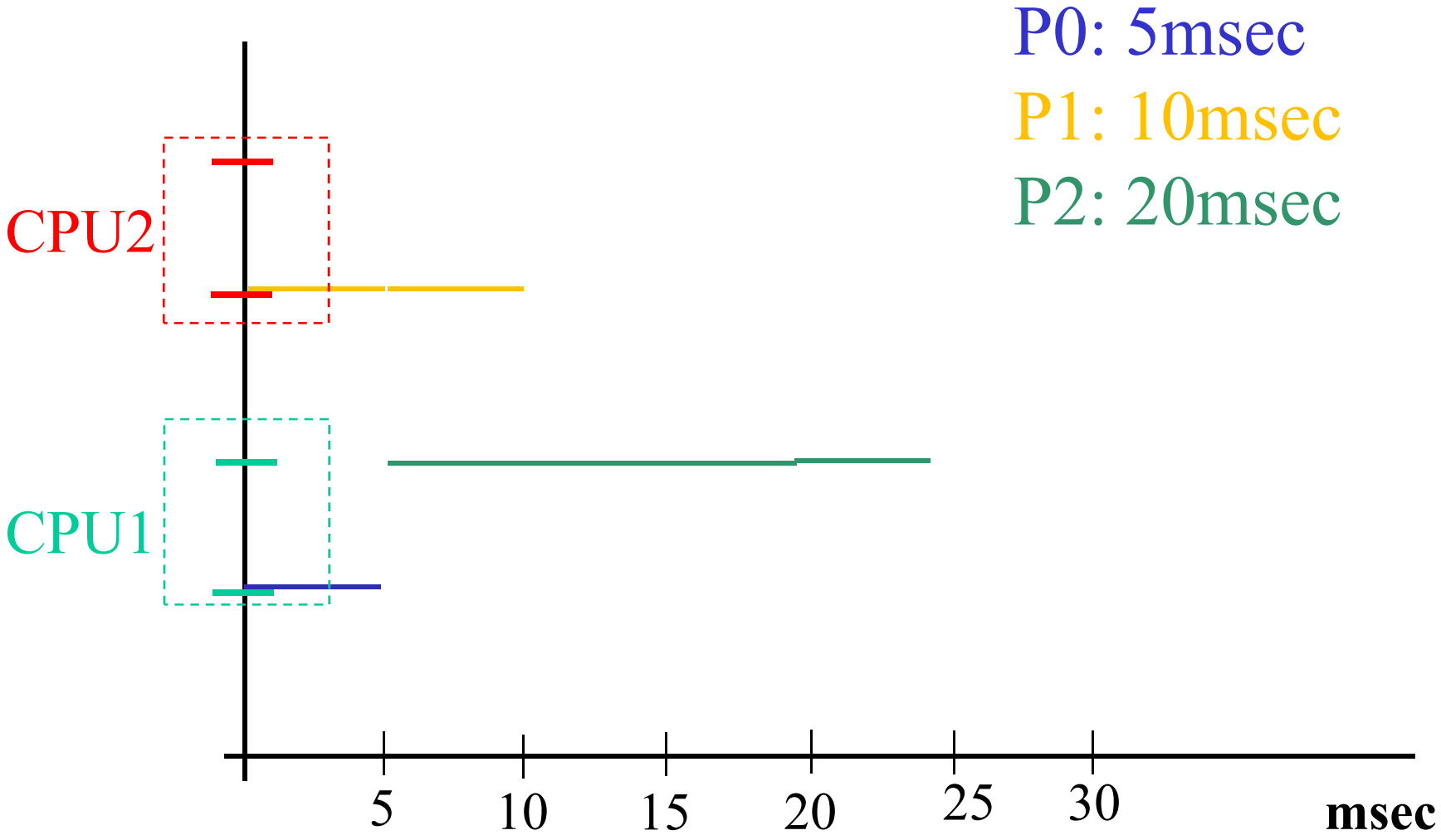
(δ) Αλλαγή του χάρτη μνήμης

Άσκηση 1.4 (Πρ. 8)

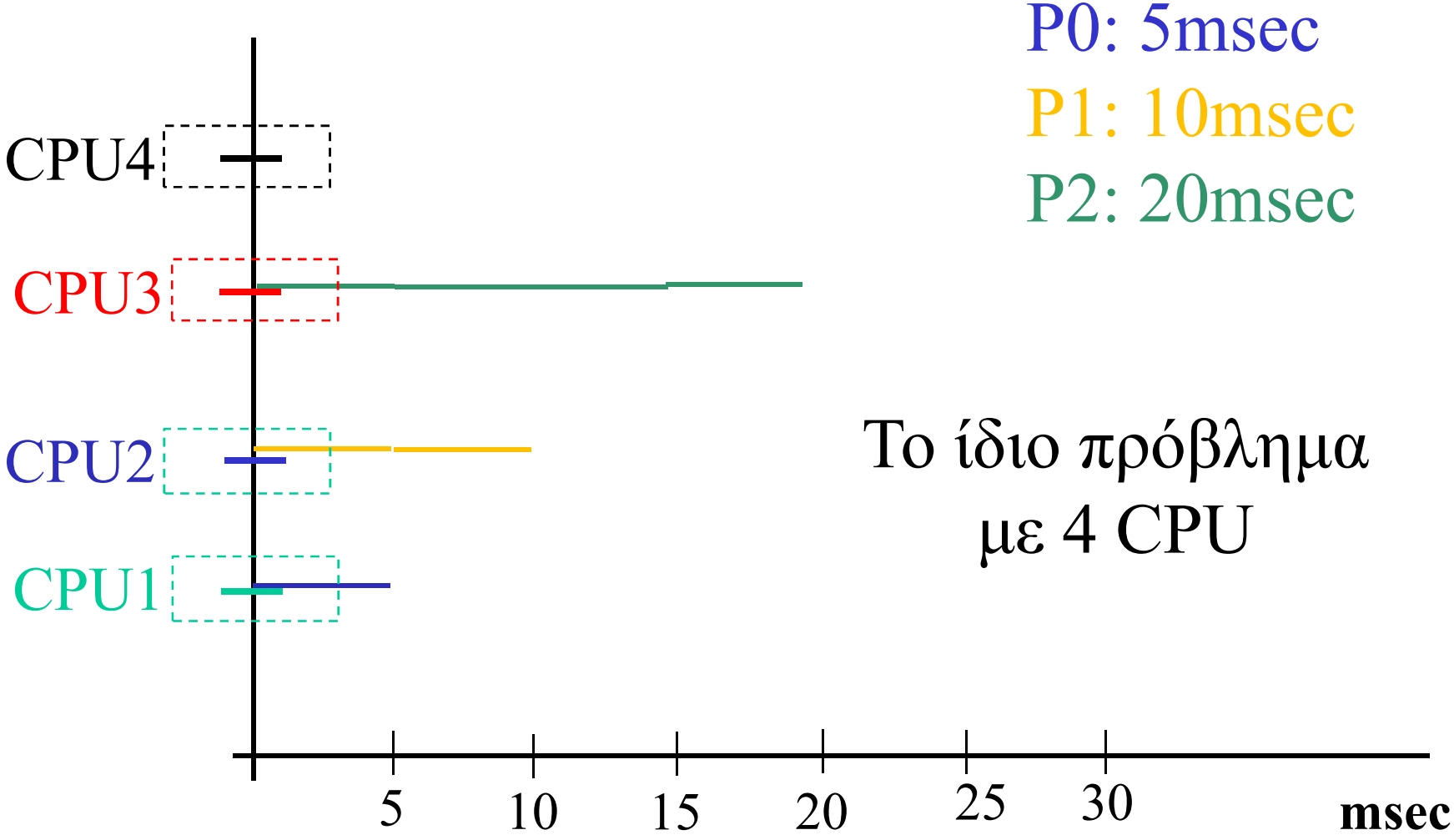
Θεωρήστε ένα σύστημα με **2 CPU**, στο οποίο κάθε CPU έχει **2** νήματα (**υπερνημάτωση – hyperthreading**). Υποθέστε ότι ξεκινούν τρία προγράμματα, τα *P0*, *P1*, και *P2*, με χρόνους εκτέλεσης 5, 10, και 20 msec αντίστοιχα. Πόσος χρόνος θα χρειαστεί για την ολοκλήρωση της εκτέλεσης και των τριών προγραμμάτων;

Υποθέστε ότι και τα 3 προγράμματα χρειάζονται κατά 100% τη CPU, δεν μπλοκάρονται κατά την εκτέλεση, και δεν αλλάζουν CPU από τη στιγμή που θα τους ανατεθεί κάποια.

Απάντηση 1.4 (Πρ. 8)



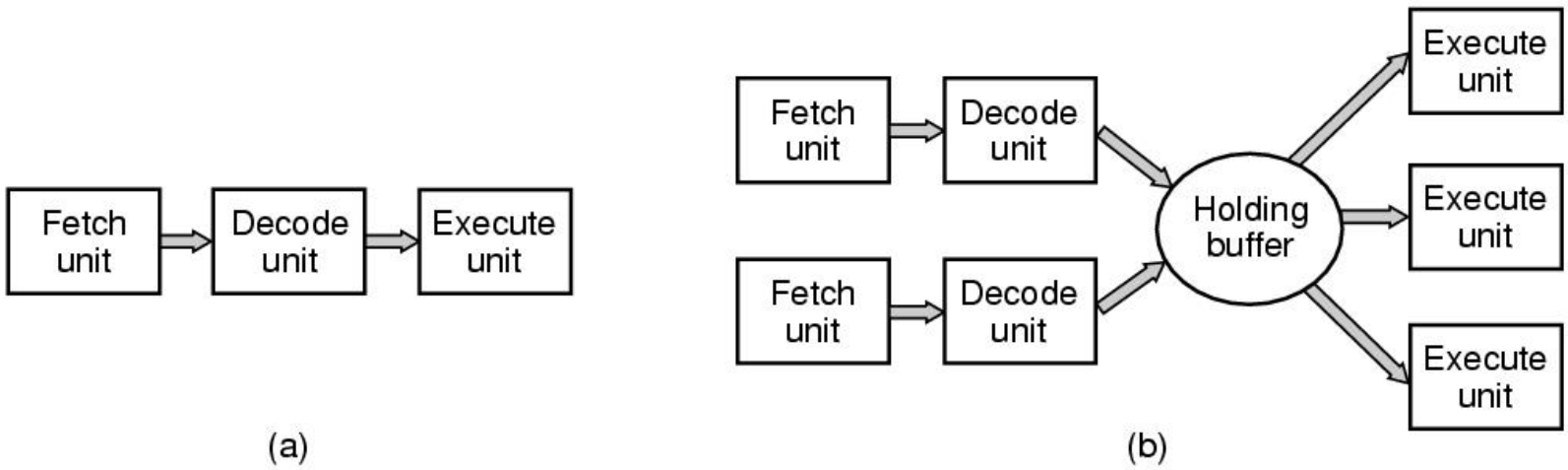
Απάντηση 1.4 (...συνέχεια)



Το ίδιο πρόβλημα
με 4 CPU

Άσκηση 1.5 (Πρ. 9)

Ένας υπολογιστής διαθέτει CPU με διοχέτευση τριών σταδίων. Κάθε στάδιο χρειάζεται τον ίδιο χρόνο για να ολοκληρώσει την εργασία του, για παράδειγμα 1 nsec. Πόσες εντολές εκτελεί αυτή η μηχανή ανά δευτερόλεπτο;



Απάντηση 1.5 (Πρ. 9)

Κάθε nsec, μία εντολή βγαίνει από τη διοχέτευση. Αυτό σημαίνει ότι η μηχανή εκτελεί 1.000.000.000 εντολές ανά δευτερόλεπτο. Δεν έχει σημασία πόσα στάδια έχει μια διοχέτευση. Μία διοχέτευση με 10 στάδια θα εκτελούσε και πάλι 1 δις εντολές ανά δευτερόλεπτο. Αυτό που έχει σημασία είναι πόσο συχνά μία **ολοκληρωμένη εντολή** βγαίνει στο τέλος της διοχέτευσης.

– Τι θα συνέβαινε εάν τα 3 στάδια είχαν χρόνο εκτέλεσης 0,5, 0,7 και 1 nsec αντίστοιχα;

Άσκηση 1.6 (Πρ. 10)

Θεωρήστε ένα σύστημα υπολογιστή που διαθέτει κρυφή μνήμη (cache memory), κύρια μνήμη (RAM) και δίσκο, και του οποίου το ΛΣ χρησιμοποιεί εικονική μνήμη (virtual memory). Χρειάζονται:

- 2 nsec για την προσπέλαση μιας λέξης από την cache memory,
- 10 nsec για την προσπέλαση μιας λέξης από τη μνήμη RAM, και
- 10 ms για την προσπέλαση μιας λέξης από το δίσκο.

Αν το ποσοστό ευστοχίας της κρυφής μνήμης είναι 95% και το ποσοστό ευστοχίας της κύριας μνήμης (μετά από μια αστοχία κρυφής μνήμης) είναι 99%, ποιος είναι ο μέσος χρόνος προσπέλασης μιας λέξης;

Απάντηση 1.6 (Πρ. 10)

1 msec = 10^{-3} sec, 1 μsec = 10^{-6} sec, 1 nsec = 10^{-9} sec

Εφόσον στο 95% πετυχαίνει η cache memory, υπολείπεται 5% για τις άλλες δύο περιπτώσεις.

Η RAM έχει ευστοχία 99%, άρα στο υπολειπόμενο 1% οδηγούμαστε στο δίσκο.

2 nsec×95% + 5% [(10 nsec×99%) + (10 ms×1%)] =

1,9 nsec + 0,495 nsec + 0,005 ms =

2,395 nsec + 5 **μ**sec = 2,395 nsec + 5×10^3 **n**sec =

= 5.002,395nsec

Άσκηση 1.7 (Πρ. 13-14)

(α) Τι είναι μια εντολή παγίδευσης (trap); Εξηγήστε τη χρήση της στα λειτουργικά συστήματα.

(β) Ποιά είναι η κυριότερη διαφορά ανάμεσα στις παγιδεύσεις και τις διακοπές (interrupts);

Απάντηση 1.7 (Πρ. 13-14)

(α) Μία παγίδευση προκαλείται από μία κλήση συστήματος (system call) ώστε να επιτρέψει σε ένα πρόγραμμα να αλλάξει την κατάσταση της εκτέλεσής του από την κατάσταση χρήστη (user mode) σε κατάσταση πυρήνα (kernel mode). Με αυτό τον τρόπο μπορεί να εκτελέσει λειτουργίες που απαιτούν ειδικά δικαιώματα (π.χ. Ε/Ε)

(β) Η παγίδευση (trap) προκαλείται από το ίδιο το πρόγραμμα, σε συγκεκριμένο(α) σημείο(α) του κώδικα. Συνεπώς, όσες φορές και εάν εκτελεστεί το πρόγραμμα, η παγίδευση θα συμβαίνει ακριβώς στην ίδια εντολή. Μία διακοπή (interrupt) προκαλείται από ένα εξωτερικό γεγονός (π.χ. ο χρονοπρογραμματιστής του ΛΣ) και οι χρονικές στιγμές που συμβαίνει δεν είναι προβλέψιμες.

Άσκηση 1.8 (Πρ. 17)

Ποιός είναι ο σκοπός των κλήσεων συστήματος σε ένα λειτουργικό σύστημα;

Απάντηση 1.8 (Πρ. 17)

Το ΛΣ είναι το μόνο λογισμικό το οποίο εκτελείται σε κατάσταση πυρήνα (kernel mode). Όλο το υπόλοιπο λογισμικό εκτελείται σε κατάσταση χρήστη (user mode).

Για να χρησιμοποιήσει μία διεργασία κάποια υπηρεσία του ΛΣ (π.χ. ανάγνωση από αρχείο) εκτελεί την αντίστοιχη κλήση συστήματος (system call).

Η κλήση συστήματος προκαλεί μία παγίδευση (trap) και μεταβαίνει σε κατάσταση πυρήνα. Η εντολή TRAP προκαλεί την ανάμιξη του ΛΣ.

Κάθε κλήση συστήματος υλοποιείται μέσω μίας ρουτίνας που βρίσκεται σε μία βιβλιοθήκη διαδικασιών (συνήθως σε C).

Άσκηση 1.9 (Πρ. 18)

Για κάθε μία από τις επόμενες κλήσεις συστήματος, δώστε μια συνθήκη που τις αναγκάζει να αποτύχουν: fork, exec, και unlink.

Απάντηση 1.9 (Πρ. 18)

Η κλήση συστήματος **fork** μπορεί να αποτύχει αν δεν υπάρχουν ελεύθερες θέσεις στον πίνακα διεργασιών (process table) (άρα δεν υπάρχει διαθέσιμος χώρος για μία νέα διεργασία).

Η κλήση συστήματος **exec** μπορεί να αποτύχει αν το όνομα του αρχείου που έχει δοθεί δεν υπάρχει ή εάν δεν αντιστοιχεί σε ένα έγκυρο εκτελέσιμο αρχείο.

Η κλήση συστήματος **unlink** μπορεί να αποτύχει αν το αρχείο προς αποσύνδεση δεν υπάρχει ή εάν η καλούσα διεργασία δεν έχει εξουσιοδότηση για να το αποσυνδέσει (π.χ. ανήκει σε άλλο χρήστη).

Άσκηση 1.10 (Πρ. 19)

Τι επιστρέφει η παρακάτω κλήση συστήματος;

counter = write(fd, &buffer, nbytes);

Μπορεί να επιστρέψει διαφορετική τιμή; Αν ναι, για ποιο λόγο;

Απάντηση 1.10 (Πρ. 19)

Επιστρέφει τον αριθμό των byte που πραγματικά αντιγράφτηκαν. Υπό κανονικές συνθήκες, επιστρέφει την τιμή nbytes.

Εάν η κλήση αποτύχει, για παράδειγμα επειδή ο περιγραφέας του αρχείου **fd** είναι λανθασμένος, τότε θα επιστρέψει την τιμή -1.

Μπορεί επίσης να αποτύχει γιατί ο δίσκος είναι γεμάτος και δεν είναι δυνατό να εγγράψει τον αριθμό των bytes που έχουν ζητηθεί.

Άσκηση 1.11 (Πρ. 20)

Ένα αρχείο που διαθέτει τον περιγραφέα αρχείου *fd* περιέχει την επόμενη ακολουθία bytes

3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5.

Γίνονται οι επόμενες κλήσεις συστήματος:

```
lseek(fd, 3, SEEK_SET);
read(fd, &buffer, 4);
```

Τι θα περιέχει η μεταβλητή *buffer* μετά από το πέρας της ανάγνωσης;

Απάντηση 1.11 (Πρ. 20)

Η κλήση lseek(fd, 3, SEEK_SET); μετακινεί το δείκτη του αρχείου κατά 3 byte από την αρχή του αρχείου (SEEK_SET).

Στη συνέχεια η κλήση read(fd, &buffer, 4); διαβάζει 4 byte από το ίδιο αρχείο (με περιγραφέα fd) και τα γράφει στο buffer.

Θα περιέχει τα bytes: 1, 5, 9, 2.

Απάντηση 1.11 (...συνέχεια)

Η κλήση lseek( );

lseek(fd, 3, SEEK_SET); /**μετακίνηση 3 byte από την αρχή*

lseek(fd, 4, SEEK_CUR); /* *μετακίνηση από την τρέχουσα θέση*

lseek(fd, -5, SEEK_END); /**μετακίνηση 5 byte πίσω, αρχίζοντας από το τέλος*

Περισσότερες πληροφορίες για τη σύνταξη όλων των κλήσεων συστήματος μπορείτε να βρείτε στο:

http://kernel.org/doc/man-pages/

Άσκηση 1.12 (Πρ. 21)

Ένα αρχείο μεγέθους 10 MB είναι αποθηκευμένο σε ένα δίσκο στην ίδια τροχιά (τροχιά 50) σε διαδοχικούς τομείς. Ο βραχίονας του δίσκου βρίσκεται προς το παρόν πάνω από την τροχιά 100. Πόσος χρόνος χρειάζεται για την ανάγνωση όλου του αρχείου από το δίσκο;

Υποθέσεις: (1) Μετακίνηση του βραχίονα από ένα κύλινδρο στον επόμενο: 1ms.

(2) Περιστροφή του τομέα που περιέχει την αρχή του αρχείου κάτω από την κεφαλή: 5ms.

(3) Ταχύτητα ανάγνωσης: 100 MB/sec.

Απάντηση 1.12 (Πρ. 21)

Μετακίνηση από track 100 → track 50:

50× 1ms = 50 ms

Περιστροφή του track 100 στην αρχή του αρχείου: 5ms

Χρόνος ανάγνωσης 10 MB με ταχύτητα 100 MB/sec:

10 MB /(100 MB/sec) = 0,1 sec = 0,1 × 10^3 ms =100ms

Συνεπώς συνολικός χρόνος: (50+5+100) ms = 155ms

Άσκηση 1.13 (Πρ. 23)

Στο παράδειγμα που δόθηκε στην εικόνα 1-17, η διαδικασία βιβλιοθήκης ονομάζεται **read** και η αντίστοιχη κλήση συστήματος ονομάζεται επίσης *read*. Είναι απαραίτητο να έχουν το ίδιο όνομα; Αν όχι, ποιά από τις δύο είναι πιο σημαντική;

Απάντηση 1.13 (Πρ. 23)

Οι κλήσεις συστήματος στην πραγματικότητα δεν έχουν ονόματα. Το όνομα μίας κλήσης συστήματος έχει νόημα μόνο για τον προγραμματιστή και όχι για το ΛΣ.

Όταν η διαδικασία βιβλιοθήκης **read** καλεί την εντολή TRAP, βάζει τον αριθμό της κλήσης συστήματος *read* σε έναν καταχωρητή (register) ή στη στοίβα (stack). Αυτός ο αριθμός χρησιμοποιείται ως δείκτης σε ένα πίνακα διεργασιών. Στην πραγματικότητα, δεν υπάρχει πουθενά ένα όνομα για κάποια κλήση συστήματος.

Από την άλλη, το όνομα της διαδικασίας βιβλιοθήκης **read** (όπως και κάθε διαδικασίας βιβλιοθήκης) είναι πολύ σημαντικό, καθώς είναι αυτό το οποίο εμφανίζεται στο πρόγραμμα.