Θέμα 1 (5)
Γιατί υπάρχει ανάγκη διαχωρισμού ανάμεσα σε user και system mode σε ένα σύστημα Η/Υ, και που χρησιμοποιείται αυτή η διαφοροποίηση σε ένα λειτουργικό σύστημα; Ποιός ο ρόλος των κλήσεων συστήματος; Εξηγήστε, χρησιμοποιώντας και κατάλληλα διαγράμματα, πως λειτουργεί και τι γίνεται κατά την διάρκεια μιας κλήσης συστήματος.

Θέμα 2 (5)
Γιατί υπάρχουν διακοπές σε ένα σύστημα Η/Υ; Ποιό πρόβλημα συγχρονισμού δημιουργούν οι διακοπές, αλλά και γιατί γίνεται προσπάθεια να ελαχιστοποιηθούν τα τμήματα του κώδικα όπου οι διακοπές είναι απενεργοποιημένες;

Θέμα 3 (10)
Έστω μια δομή στην μνήμη του συστήματος που διαβάζεται και μεταβάλλεται από πολλές διεργασίες ταυτόχρονα. Δώστε τον κώδικα συγχρονισμού (κώδικες εισόδου/εξόδου) για τον έλεγχο πρόσβασης σε αυτή την δομή, έτσι ώστε (α) να μην επιτρέπεται η ταυτόχρονη ανάγνωση και μεταβολή των δεδομένων, (β) να μην επιτρέπεται η ταυτόχρονη μεταβολή δεδομένων, και (γ) η ταυτόχρονη ανάγνωση να περιορίζεται στο πολύ n διεργασίες. Χρησιμοποιήστε σημαφόρους.

Θέμα 4 (10)
Δίνεται το εξής σύστημα ηλεκτρονικού ταχυδρομείου. Η διεργασία «αποστολέας» τοποθετεί τον μήνυμα σε μια ουρά FIFO με πεπερασμένο μέγεθος Ν. Στην συνέχεια, διεργασίες «μεταδότες» απομακρύνουν τα μηνύματα, ένα προς ένα, από την ουρά και τα στέλνουν πάνω από το διαδίκτυο. Δώστε τον κώδικα τοποθέτησης και απομάκρυνσης μηνυμάτων έτσι ώστε οι διεργασίες «αποστολείς» να μπλοκάρονται όσο δεν υπάρχει χώρος στην ουρά ενώ οι διεργασίες «μεταδότες» να μπλοκάρονται όσο η ουρά είναι άδεια. Χρησιμοποιήστε monitors και conditions. Υποθέστε πως η ουρά είναι αφηρημένος τύπος δεδομένων με πράξεις: void init(msgq *), void put(msgq *, msg), msg get(msgq *), για την αρχικοποίηση, και αντίστοιχα τοποθέτηση και απομάκρυνση ενός μηνύματος.

Θέμα 5 (5)
Δίνεται ο εξής μηχανισμός δέσμευσης πόρων. Κάθε διεργασία έχει ένα διάνυσμα στο οποίο καταγράφεται ο αριθμός των πόρων που περιμένει η διεργασία να λάβει για να συνεχίσει την εκτέλεση της. Αιτήσεις για δέσμευση πόρων καθώς και απελευθέρωση πόρων είναι επιτρεπτή ανά πάσα στιγμή. Όταν μια διεργασία P1 κάνει μια αίτηση για ένα πόρο R, και δεν υπάρχει ελεύθερη οντότητα του ζητούμενου πόρου, τότε το σύστημα βρίσκει μια μπλοκαρισμένη διεργασία P2 στην οποία έχει δοθεί μια οντότητα του πόρου R, αφαιρεί τον πόρο από αυτή την διεργασία P2 (το διάνυσμα της αυξάνεται κατά 1 για τον πόρο R) και δίνει τον πόρο στην διεργασία P1. Αν δεν υπάρχει μπλοκαρισμένη διεργασία που να κρατά μια οντότητα του ζητούμενου πόρου R, τότε η διεργασία P1 μπλοκάρεται. Για παράδειγμα, έστω πως το σύστημα αρχικά διαθέτει ελεύθερους πόρους (Α=4,Β=2,Γ=2). Η διεργασία P1 κάνει αίτηση (2,2,1) και λαμβάνει τους πόρους. Στη συνέχεια, η διεργασία P2 κάνει αίτηση (1,0,1) και λαμβάνει και αυτή τους πόρους. Στη συνέχεια η διεργασία P1 κάνει νέα αίτηση (0,0,1) και μπλοκάρεται, οπότε και το διάνυσμα της γίνεται (0,0,1). Η διεργασία P3 κάνει αίτηση (2,0,0), και λαμβάνει τον υπολοιπόμενο ελεύθερο πόρο Α καθώς και άλλο ένα πόρο τύπου Α από την διεργασία P1 που έχει μπλοκαριστεί, και το διάνυσμα της μπλοκαρισμένης διεργασίας P1 διαμορφώνεται πλέον σε (1,0,1). Μπορεί αυτός ο μηχανισμός να οδηγήσει σε αδιέξοδο; Αν ναι δώστε ένα παράδειγμα, διαφορετικά εξηγήστε ποιά συνθήκη αδιεξόδου δεν μπορεί να ικανοποιηθεί ποτέ με αυτό το μηχανισμό.

Θέμα 6 (10)
Τρείς διεργασίες Α, Β, και Γ, καταφθάνουν την χρονική στιγμή 0.0, 0,5 και 1.0 αντίστοχα ενώ ο χρόνος εκτέλεσης (CPU burst time) είναι 8, 4, και 1 ms αντίστοιχα. Ποιός ο μέσος χρόνος διεκπεραίωσης για αυτές τις διεργασίες αν χρησιμοποιηθούν οι αλγόριθμοι (α) FCFS και (β) SJF; Σχεδιάστε τα χρονοδιαγράμματα Gantt που δείχνουν την εκτέλεση των διεργασιών για κάθε έναν από τους παραπάνω αλγόριθμους. Ποιός θα ήταν ο μέσος χρόνος διεκπεραίωσης για την περίπτωση του SJF, αν το σύστημα περίμενε για 1 μονάδα χρόνου πριν αρχίσει να εξυπηρετεί τις διεργασίες; Πως μπορεί ένα σύστημα χρονοπρογραμματισμού να «επιδιορθώσει» το γεγονός πως δεν μπορεί να προβλέψει το μέλλον;

Θέμα 7 (10)

Έστω ένα σύστημα με διευθύνσεις 32 bit, φυσική μνήμη 256Mbyte και πλαίσια 4Kbyte, και έστω πως κάθε διεργασία στο σύστημα πρέπει να έχει εικονική μνήμη 1Gbyte. Ορίστε ένα λογικό σχήμα σελιδοποίησης διπλού επιπέδου, δίνοντας την δομή των εικονικών διευθύνσεων. Επίσης, δώστε το επιπλέον κόστος σε μνήμη ανά διεργασία που συνεπάγεται το σχήμα σας για το σύστημα, αν υποθέσετε πως για να τρέξει μια διεργασία ο πίνακας πρώτου επιπέδου και τουλάχιστον ένας πίνακας δευτέρου επιπέδου πρέπει να βρίσκονται στην κυρίως μνήμη. Ποιό θα ήταν αυτό το κόστος αν χρησιμοποιούσατε σελιδοποίηση ενός επιπέδου;

Θέμα 8 (5)
Εξηγήστε (χρησιμοποιώντας και διαγράμματα) πως με χρήση εικονικής μνήμης μπορούν δύο ή περισσότερες διεργασίες να αναφέρονται στα ίδια δεδομένα στην φυσική μνήμη. Δώστε τουλάχιστον δύο διαφορετικές περιπτώσεις που αυτό είναι ιδιαίτερα χρήσιμο σε επίπεδο λειτουργικού συστήματος και εξηγήστε γιατί.

Θέμα 9 (5)
Σε ένα σύστημα εικονικής μνήμης με σελιδοποίηση και χρήση δίσκου για την αντικατάσταση σελίδων μετρήθηκαν τα εξής: φόρτος ΚΜΕ 20%, φόρτος δίσκου για εικονική μνήμη 97.7%, φόρτος υπολοίπων συσκευών I/O 5%. Τι νομίζετε πως συμβαίνει στο σύστημα; Ποιά από τα παρακάτω θα βελτίωναν πιθανώς την απόδοση του συστήματος: (α) πιο γρήγορη ΚΜΕ, (β) μεγαλύτερος δίσκος, (γ) αύξηση βαθμού πολυπρογραμματισμού, (δ) περισσότερη μνήμη, (ε) εγκατάσταση πολλών δίσκων με παράλληλη δρομολόγηση, (στ) αύξηση μεγέθους σελίδας. Τεκμηριώστε τις απαντήσεις σας.

Θέμα 10 (10)
Θεωρείστε την ακόλουθη συμβολοσειρά αναφορών σε σελίδες:

<1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6>

Πόσα σφάλματα σελίδας θα συμβούν με τους αλγόριθμους αντικατάστασης σελίδων LRU, FIFO και Optimal, υποθέτοντας ότι το σύστημα διαθέτει τέσσερα πλαίσια; Υποθέστε ότι τα πλαίσια αυτά αρχικά είναι άδεια.

Θέμα 11 (5)
Εξηγήστε (και με παραδείγματα) ποιοί από τους αλγόριθμους χρονοπρογραμματισμού δίσκου μπορεί να οδηγήσουν σε «άδικη μεταχείριση» διεργασιών σε ένα πολυπρογραμματιστικό σύστημα: (α) FCFS, (β) SSTF, και (γ) C-SCAN. Εξηγήστε γιατί η «άδικη μεταχείριση» δεν είναι επιθυμητή, αλλά και γιατί η «απόλυτη δικαιοσύνη» στον χρονοπρογραμματισμό δίσκου σε ένα πολυπρογραμματιστικό σύστημα δεν είναι ιδιαίτερα πρακτική.

Θέμα 12 (5)
Ο κατακερματισμός στον δίσκο μπορεί να επιδιορθωθεί μεταφέροντας τα δεδομένα των αρχείων σε συνεχόμενες περιοχές του δίσκου. Εξηγήστε τι αυτό συνεπάγεται αν το σύστημα αρχείων χρησιμοποιεί: (α) συνεχόμενη αποθήκευση, (β) συνδεδεμένη αποθήκευση, και (γ) ευρετηριακή αποθήκευση. Δώστε 3 λόγους γιατί τελικά αυτό δεν γίνεται (συχνά) στα μοντέρνα λειτουργικά συστήματα.