Κεφάλαιο 13

Λειτουργικά Συστήματα

13.1 Εισαγωγή

Κάθε υπολογιστικό σύστημα αποτελείται από το υλικό (hardware) και το λογισμικό (software). Με τον όρο υλικό αναφερόμαστε στο σύνολο των συσκευών όπως Κεντρική Μονάδα επεξεργασίας, μονάδες αποθήκευσης, μονάδες εισόδου-εξόδου (εκτυπωτές, πληκτρολόγιο, οθόνη). Με τον όρο λογισμικό αναφερόμαστε τόσο στο λειτουργικό σύστημα όσο και στα προγράμματα εφαρμογής που γράφει ο χρήστης προκειμένου να δώσει εντολές στον υπολογιστή για τον τρόπο χρησιμοποίησης των συσκευών.

Λειτουργικό σύστημα είναι το σύνολο των προγραμμάτων που ασχολούνται με τη διαχείριση και τον συντονισμό των εργασιών, καθώς και την κατανομή των διαθέσιμων πόρων. Το λειτουργικό σύστημα ουσιαστικά αποτελεί το σύνδεσμο μεταξύ λογισμικού και υλικού του υπολογιστικού συστήματος. Μια από τις βασικές αρμοδιότητες του λειτουργικού συστήματος είναι η διαχείριση του υλικού. Με αυτό τον τρόπο οι εφαρμογές του χρήστη απλουστεύονται εφόσσον απαλλάσονται από τον άμεσο και επίπονο χειρισμό του υπολογιστή και παρέχεται έτσι η δυνατότητα σε χρήστες με περιορισμένες γνώσεις υπολογιστών να αναπτύσουν τις εφαρμογές τους. Αν για παράδειγμα δεν υπήρχε το λειτουργικό σύστημα θα όφειλε ο προγραμματιστής να γράφει πρόσθετο κώδικα ο οποίος θα παρακολουθούσε τακτικά αν υπάρχει κάποια είσοδος από το πληκτρολόγιο, θα έγραφε κώδικα για να στείλει κάτι προς εκτύπωση και ουτω καθεξής. Αρμοδιότητα επίσης του λειτουργικού συστήματος αποτελεί η βέλτιστη χρήση του υλικού έτσι ώστε να αυξάνεται η αποδοτικότητα του υπολογιστικού συστήματος.

13.2 Ιστορική Αναδρομή

Τη δεκαετία του ’40 παρουσιάστηκαν οι πρώτοι Η/Υ οι οποίοι βασίζονταν στις λυχνίες κενού και δεν διέθεταν λειτουργικό σύστημα (λειτουργικά συστήματα μηδενικής γενιάς). Οι χρήστες των πρώτων Η/Υ ήταν μόνο έμπειροι προγραμματιστές οι οποίοι έδιναν εντολές χειριζόμενοι τους διακόπτες και τα σήματα ελέγχου είχαν άμεση προσπέλαση στη γλώσσα μηχανής και προγραμμάτιζαν τα πάντα διότι δεν υπήρχε λειτουργικό σύστημα να αναλάβει τη διαχείριση του

υλικού. Την περίοδο εκείνη ένας άνθρωπος ή μία ομάδα ανθρώπων σχεδίαζε, υλοποιούσε, προγραμμάτιζε και συντηρούσε τον υπολογιστή. Αυτό που επι το πλείστον έκαναν αυτοί οι πρώτοι υπολογιστές ήταν αριθμητικοί υπολογισμοί (ημίτονο, συνημίτονο, λογάριθμος) και ο τρόπος που γινόταν ήταν με καλωδίωση των πινάκων συνδέσεων και σε γλώσσα μηχανής δεδομένου ότι δεν υπήρχαν οι γλώσσες προγραμματισμού. Το να καεί δε μία λυχνία κατά τη διάρκεια της εκτέλεσης σήμαινε επανάληψη της διαδικασίας από την αρχή.

Στα μέσα της δεκαετίας του ’50 η χρήση των τρανζίστορ κάνει τους υπολογιστές πιο αξιόπιστους. Δεν υπήρχε σοβαρός κίνδυνος να διακοπεί η εκτέλεση μιας εργασίας και να χρειαστεί να ξεκινήσει από την αρχή. Αυτοί οι υπολογιστές που είναι γνωστοί ως mainframes τοποθετούνταν σε μεγάλους κλιματιζόμενους χώρους και προσωπικό διαφορετικών ειδικοτήτων απασχολούνταν με αυτούς. Οι πελάτες που αγόραζαν αυτούς τους υπολογιστές ήταν μεγάλες εταιρείες, κρατικές υπηρεσίες και πανεπιστήμια. Οι λόγοι μεταξύ άλλων ήταν το μέγεθός τους και το κόστος τους. Ο τρόπος που γίνονταν οι εργασίες διέφερε αρκετά από την προηγούμενη γενιά. Καταρχάς υπήρχε εξειδίκευση στο προσωπικό που χρησιμοποιούσε τους υπολογιστές και έτσι είχαμε σχεδιαστές, κατασκευαστές, χειριστές, προγραμματιστές και προσωπικό συντήρησης.Έτσι αρχικά κάποιος έγραφε σε χαρτί το πρόγραμμα που επίλυε κάποιο πρόβλημα. Η γλώσσα που χρησιμοποιούσε ήταν assembly ή fortran. Στη συνέχεια ο κώδικας γραφόταν σε διάτρητες κάρτες. Οι χεριστές έπαιρναν τις κάρτες και τις εισήγαγαν στον υπολογιστή για ανάγνωση, ακολουθούσε η εκτέλεση του κώδικα και ο χειριστής έπαιρνε τα αποτελέσματα που τυπωνόταν σε κάποιον εκτυπωτή και συνήθως δεν βρισκόταν στον ίδιο χώρο. Επειδή υπήρχε μεγάλη καθυστέρηση σε αυτή τη διαδικασία υιοθετήθηκε το σύστημα δέσμης (batch system). Στο σύστημα αυτό ένα πλήθος καρτών με κώδικα δέσμης εργασιών συγκεντρωνόταν (σε χρονικό διάστημα μιας ώρας για παράδειγμα) και ένας υπολογιστής χαμηλού κόστους διάβαζε τις κάρτες και τις έγραφε σε μαγνητική ταινία. Η ταινία με τα προγράμματα καθώς και το πρόγραμμα που θα μπορούσε να χαρακτηριστεί ως κώδικας λειτουργικού συστήματος φορτωνόταν στον υπολογιστή που εκτελούσε τις εργασίες σειριακά και στη συνέχεια έγραφε τα αποτελέσματα σε ταινία και όχι στον εκτυπωτή. Όταν ολοκληρωνόταν και η τελευταία εργασία ο χειριστής τοποθετούσε την ταινία με τα αποτελέσματα σε έναν εκτυπωτή ο οποίος δεν ήταν συνδεδεμένος με τον υπολογιστή και έτσι ο υπολογιστής έπαιρνε την επόμενη δέσμη εργασιών. Έτσι έχουμε τα λειτουργικά συστήματα της πρώτης γενιάς που ονομάζονται Λειτουργικά Συστήματα ομαδικής επεξεργασίας (batch processing).

Τα λειτουργικά συστήματα πρώτης γενιάς δέχονταν τη δέσμη με τα προγράμματα που υπέβαλαν οι χρήστες και τα επεξεργαζόταν το ένα μετά το άλλο με τη σειρά. Βασικό μειονέκτημα αυτών των λειτουργικών ήταν η υποαπασχόληση των συσκευών. Τα προγράμματα μπορούσαν να εκτελεστούν σειριακά και καμία ταυτόχρονη εκτέλεση δεν ήταν εφικτή. Έτσι δεδομένου ότι ένα πρόγραμμα δεν θα μπορούσε να απασχολεί ταυτόχρονα όλες τις συσκευές υλικού αναγκαστικά όλες οι υπόλοιπες συσκευές παρέμεναν ανενεργές. Ο χρόνος ανακύκλωσης ενός προγράμματος σε αυτά τα λειτουργικά συστήματα θα μπορούσε να είναι 50-60 φορές μεγαλύτερος από τον χρόνο εκτέλεσης του συγκεκριμένου προγράματος. Ως χρόνος ανακύκλωσης ορίζεται ο χρόνος από τη στιγμή της υποβολής ενός προγράμματος έως τη στιγμή λήψης των αποτελεσμάτων. Οι υπολογιστές της γενιάς αυτής χρησιμοποιήθηκαν κυρίως για επιστημονικούς υπολογισμούς όπως η επίλυση μερικών διαφορικών εξισώσεων. Η γλώσσα προγραμματισμού είναι η Fortran.

Στη δεκαετία του ’60 έχουμε τη δεύτερη γενιά λειτουργικών συστημάτων. Τα συστήματα αυτά ονομάζονται Λετουργικά Συστήματα Πολυπρογραμματισμού (multiprogramming). Στα συστή-

ματα αυτά μειώθηκε σημαντικά ο άεργος χρόνος των συσκευών του υλικού και αυτό είχε ως συνέπεια την μείωση του χρόνου ανακύκλωσης. Θα μπορούσε να ειπωθεί πως με τα συστήματα αυτά υπήρχε ένα είδος παράλληλης επεξεργασίας των προγραμμάτων που υποβάλλονται σε κάποιο υπολογιστή αλλά αυτή η παραλληλία είναι εικονική. Ο επεξεργαστής μία λειτουργία εκτελεί κάθε φορά αλλά διακόπτεται με τη χρήση σημάτων διακοπής (interrupts) προκειμένου να επιτευχθεί η βέλτιστη δυνατή χρήση των συσκευών. Έτσι ένα πρόγραμμα μπορεί να εκτελείται στην Κεντρική Μονάδα Επεξεργασίας, ένα άλλο μπορεί να διαβάζει από το σκληρό δίσκο, ένα άλλο να στέλνει δεδομένα στη συσκευή εξόδου και ουτω καθεξής. Το βασικό μειονέκτημα των λειτουργικών συστημάτων της δεύτερης γενιάς σε σχέση με αυτά της προηγούμενης είναι πως το λειτουργικό σύστημα γίνεται πλέον ένα πιο πολύπλοκο πρόγραμμα που απασχολεί και αυτό τον υπολογιστή εις βάρος των προς εκτέλεση προγραμμάτων.

Ακολουθούν τα λειτουργικά συστήματα καταμερισμού χρόνου. Σε αυτά κάθε χρήστης πληκτρολογεί τον κώδικά του σε ένα τερματικό και ένα υπολογιστικό σύστημα αναλαμβάνει την εκτέλεση των προγραμμάτων των χρηστών. Οι χρήστες έχουν την εντύπωση πως εξυπηρετούνται παράλληλα αλλά αυτό που γίνεται ουσιαστικά είναι ο καταμερισμός χρόνου του υπολογιστικού συστήματος. Το υπολογιστικό σύστημα δηλαδή καταμερίζει το χρόνο του εκ περιτροπής σε κάθε χρήστη. Το ρόλο του ρολογιού εδώ έχει ένα κύκλωμα που ονομάζεται timer (χρονιστής) και το οποίο σε τακτά χρονικά διαστήματα στέλνει σήμα διακοπής. Σε περίπτωση που κάποιο πρόγραμμα πρέπει να εκτελεστεί χωρίς καθυστέρηση δίνεται προτεραιότητα σε αυτό και εφόσσον ολοκληρωθεί η εκτέλεσή του συνεχίζεται ο καταμερισμός του χρόνου στα υπόλοιπα προγράμματα.Η συνεχής εξέλιξη του υλικού καθώς και η ανάγκη για μεγαλύτερες ταχύτητες οδήγησε στην ανάπτυξη πιο εξελιγμένων και πιο πολύπλοκων λειτουργικών συστημάτων. Οι παράλληλοι και κατανεμημένοι υπολογιστές καθώς και τα δίκτυα υπολογιστών ήταν εκείνα που δημιούργησαν τις νέες ανάγκες για αποδοτικότερα λειτουργικά συστήματα. Στα συστήματα αυτά γίνεται πραγματική και όχι εικονική παράλληλη επεξεργασία εφόσσον κάθε επεξεργαστής ασχολείται με διαφορετικό πρόγραμμα.

Στα παράλληλα συστήματα έχουμε πολλές κεντρικές μονάδες επεξεργασίας στο ίδιο μηχάνημα. Με αυτό τον τρόπο επιτυγχάνεται η παράλληλη (ταυτόχρονη) επεξεργασία πολλών προγραμμάτων εφόσσον δεν είναι αναγκαίο να εξυπηρετηθούν όλα από την ίδια ΚΜΕ. Στα κατανεμημένα συστήματα οδήγησε η δικτύωση και η διαδικτύωση. Μία εργασία μπορεί πλέον να διεκπεραιωθεί σε έναν υπολογιστή που βρίσκεται πολλά χιλιόμετρα μακριά. Υπάρχει επίσης η δυνατότητα να εκτελείται τμήμα του κώδικα σε έναν υπολογιστή, άλλο τμήμα του σε άλλο υπολογιστή αρκεί να υπάρχει δικτύωση μεταξύ τους. Τα σημερινά λειτουργικά συστήματα των προσωπικών υπολογιστών υποστηρίζουν πολυπρογραμματισμό και αυτό γίνεται φανερό από την εκκίνηση του υπολογιστή όπου πολλά προγράμματα τρέχουν ταυτόχρονα. Τα πιο γνωστά εξ'αυτών είναι τα Windows, Linux, Macintosh. Η συνεχής εξέλιξη του υλικού καθώς και η ανάγκη για μεγαλύτερες ταχύτητες οδήγησε στην ανάπτυξη πιο εξελιγμένων και πιο πολύπλοκων λειτουργικών συστημάτων. Οι παράλληλοι και κατανεμημένοι υπολογιστές καθώς και τα δίκτυα υπολογιστών ήταν εκείνα που δημιούργησαν τις νέες ανάγκες για αποδοτικότερα λειτουργικά συστήματα. Στα συστήματα αυτά γίνεται πραγματική και όχι εικονική παράλληλη επεξεργασία εφόσσον κάθε επεξεργαστής ασχολείται με διαφορετικό πρόγραμμα.

Λειτουργικά συστήματα σήμερα έχουμε επίσης και στους υπολογιστές χειρός. Τα λειτουργικά συστήματα που εκτελούνται στους υπολογιστές χειρός είναι αρκετά περίπλοκα δεδομένου ότι

υπάρχουν απαιτήσεις για χειρισμό τηλεφωνίας, ψηφιακής φωτογραφίας κ.λπ. Μια άλλη κατηγορία σημερινών λειτουργικών συστημάτων είναι αυτά που χρησιμοποιούνται σε συσκευές οι οποίες δεν είναι ακριβώς υπολογιστές όπως η τηλεόραση, ο φούρνος μικροκυμάτων κ.λπ. Αυτά είναι τα ενσωματωμένα λειτουργικά συστήματα και η βασική τους διαφορά από τα λειτουργικά συστήματα χειρός είναι πως δεν θα απαιτηθεί να δώσει κάποιος χρήστης ένα πρόγραμμα για εκτέλεση. Τέλος θα πρέπει να αναφερθούν και τα λετουργικά συστήματα πραγματκού χρόνου στα οποία ο χρόνος είναι πολύ σημαντικός. Για παράδειγμα στη βιομηχανία ένα ρομπότ θα πρέπει να λειτουργήσει σωστά κάποια συγκεκριμένη χρονική στιγμή. Σε περίπτωση που αυτό δεν γίνει όταν ακριβώς πρέπει πιθανόν να καταστραφεί ολόκληρη η παραγωγή. Λειτουργικά συστήματα πραγματικού χρόνου συναντούμε στην αυτοκινητοβιομηχανία, στην βιομηχανία αεροσκαφών κ.λπ. Στις έξυπνες κάρτες επίσης εκτελούνται κάποια λειτουργικά συστήματα.

13.3 Βασικές έννοιες των Λειτουργικών Συστημάτων

Μία από τις πιο βασικές έννοιες των λετουργικών συστημάτων είναι η διεργασία. Διεργασία ονομάζεται ένα πρόγραμμα ή ένα αυτόνομο τμήμα προγράμματος που εκτελείται. Η διαφορά της διεργασίας από το πρόγραμμα είναι πως το πρόγραμμα είναι ανενεργό, ουσιαστικά είναι ένα σύνολο εντολών αποθηκευμένων στη μνήμη. Η διεργασία είναι ένα πρόγραμμα που έχει ξεκινήσει να εκτελείται αλλά δεν έχει ολοκληρωθεί. Μια διεργασία μπορεί να βρίσκεται σε μία από τις παρακάτω τρεις καταστάσεις:

Να χρησιμοποιεί την ΚΜΕ.

Να έχει διακοπεί προσωρινά για να εκτελεστεί κάποια άλλη.

Να είναι μπλοκαρισμένη γιατί περιμένει κάτι για να συνεχίσει.

Στην πρώτη περίπτωση η διεργασία εκτελείται κανονικά. Στη δεύτερη η εκτέλεσή της έχει διακοπεί προκειμένου να παραχωρηθεί η ΚΜΕ σε κάποια άλλη διεργασία η οποία θα εκτελεστεί για κάποιο χρονικό διάστημα έως ότου ολοκληρωθεί ή διακοπεί για να εκτελεστεί κάποια άλλη. Στην τρίτη περίπτωση η διεργασία ενδέχεται να περιμένει κάποια δεδομένα για παράδειγμα που δεν είναι διαθέσιμα εκείνη τη στιγμή.Ας δούμε όμως πιο πρακτικά τι σημαίνει διεργασία. Ας υποθέσουμε ότι έχουμε γράψει μία εφαρμογή σε κάποια γλώσσα προγραμματσμού και απατείται κάποιο χρονικό διάστημα για να πάρουμε τα πειραματικά αποτελέσματα. Ώσπου να ολοκληρωθεί αυτή η διαδικασία κάνουμε μια αναζήτηση στο διαδίκτυο. Παράλληλα είναι ενεργοποιημένη μία διεργασία η οποία ελέγχει αν υπάρχουν εισερχόμενα μηνύματα ηλεκτρονικού ταχυδρομείου. Τι συμβαίνει με το λειτουργικό σύστημα σε αυτή την περίπτωση; Υπάρχουν τρεις διεργασίες που εκτελούνται ταυτόχρονα: εκείνη του προγράμματος που υπολογίζει τα πειραματικά αποτελέσματα, εκείνη της αναζήτησης στο διαδίκτυο και εκείνη που ελέγχει και ενημερώνει για την ύπαρξη νέων μηνυμάτων. Το λειτουργικό σύστημα διανέμει την ΚΜΕ στις τρεις διεργασίες. Εκείνο είναι που περιοδικά αποφασίζει πότε έχει ολοκληρωθεί ο χρόνος δέσμευσης της ΚΜΕ από τη μία διεργασία και αποφασίζει να συνεχιστεί η εκτέλεση της επόμενης διεργασίας. Κάθε διεργασία διακόπτεται και συνεχίζεται όταν έρθει ξανά η σειρά της για να εκτελστεί. Θα πρέπει

όμως κάθε φορά που διακόπτεται η εκτέλεση μιας διεργασίας να κρατούνται όλες οι πληροφορίες που απαιτούνται έτσι ώστε όταν έρθει πάλι η σειρά της να χρησιμοποιήσει την ΚΜΕ να μπορεί να συνεχιστεί η εκτέλεσή της. Ας υποθέσουμε για παράδειγμα πως η διεργασία έχει ανοίξει κάποια αρχεία για διάβασμα και κάποια για εγγραφή. Όταν διακοπεί για να εκτελεστεί μια άλλη διεργασία θα πρέπει να ξέρει για κάθε αρχείο τηντρέχουσα θέση στην οποία βρισκόταν πριν ανασταλεί η εκτέλεσή της έτσι ώστε να συνεχίσει την ανάγνωση ή την εγγραφή από τη σωστή θέση. Στα περισσότερα λειτουργικά συστήματα όλες αυτές οι πληροφορίες οι σχετικές με την αναστολή εκτέλεσης της διεργασίας αποθηκεύονται στον πίνακα διεργασιών. Στον πίνακα αυτό υπάρχουν όλες οι διεργασίες των οποίων η εκτέλεση έχει ανασταλεί με τις τιμές των καταχωρητών της κ.λπ σε ένα πίνακα ή μία δομή. Οι διεργασίες "ανταγωνίζονται" η μία την άλλη στην χρήση των πόρων του υπολογιστικού συστήματος. Για παράδειγμα ενώ μια διεργασία εκτελείται κάποια άλλη περιμένει να χρησιμοποιήσει την ΚΜΕ. Τον συγχρονισμό των εργασιών τον αναλαμβάνει ο διαχειριστής διεργασιών(ή χρονοπρογραμματιστής διεργασιών) ο οποίος χρησιμοποιεί ουρές αναμονής και αποφασίζει πότε θα εκτελεστεί μία διεργασία, πότε θα διακοπεί, ποιά θα είναι η επόμενη και για πόσο χρονικό διάστημα θα εκτελείται η τρέχουσα διεργασία. Οι πληροφορίες κάθε διεργασίας αποθηκεύονται στο τμήμα ελέγχου διεργασίας που συσχετίζεται άμεσα με την διεργασία. Ο διαχειριστής διεργασιών αποθηκεύει στις ουρές αναμονής το τμήμα ελέγχου κάθε διεργασίας και όχι την ίδια τη διεργασία διότι το μέγεθός τους είναι μεγάλο ώστε να μπουν ολόκληρες στη λίστα αναμονής. Έτσι το τμήμα ελέγχου διεργασίας παραμένει στην ουρά αναμονής και η ίδια η διεργασία είναι αποθηκευμένη στον δίσκο ή στη μνήμη. Η ουρά αναμονής για την ΚΜΕ δεν είναι η μοναδική, υπάρχουν ουρές για τις συσκευές εισόδου εξόδου και μάλιστα καθε τέτοια συσκευή έχει τη δική της λίστα αναμονής, λίστα για χρήση μνήμης κ.λπ. Η επιλογή της επόμενης διεργασίας σε κάθε ουρά αποφασίζεται από τον διαχειριστή διεργασιών ανάλογα με τον τρόπο που είναι προγραμματισμένος να επιλέξει την επόμενη εργασία. Υπάρχουν διάφοροι αλγόριθμοι που συντονίζουν τον χρονοπρογραμματισμό των εργασιών λαμβάνοντας υπόψη τόσο την αποδοτικότητα του συστήματος όσο και την απονομή "δικαιοσύνης" στο χρόνο απασχόλησης της ΚΜΕ. Για παράδειγμα το σχήμα First In First Out (FIFO) επιτρέπει στην διεργασία που μπήκε πρώτη στην ουρά αναμονής να εξυπηρετηθεί πρώτη. Σε άλλες υλοποιήσεις εξυπηρετείται πρώτα η πιο σύντομη ή αυτή που έχει μεγαλύτερη προτεραιότητα κ.λπ.

13.4 Κρίσιμα Τμήματα και Αμοιβαίος Αποκλεισμός Διαδιεργασιακή επικοινωνία

Οι διεργασίες κάποιες φορές επικοινωνούν. Για παράδειγμα η μία περιμένει κάποια δεδομένα εισόδου από την έξοδο κάποιας άλλης διεργασίας. Εδώ υπάρχει το πρόβλημα του πως θα μεταφερθούν τα δεδομένα από τη μία διεργασία στην άλλη καθώς και πως θα γίνει ο συγχρονισμός των διεργασιών με τέτοιο τρόπο που να μπορεί η διεργασία που περιμένει δεδομένα να μην εκτελεστεί εως ότου τα λάβει. Επίσης πολλές φορές συμβαίνει σε κάποιο υπολογιστικό σύστημα να υπάρχουν πόροι οι οποίοι μοιράζονται. Για παράδειγμα τι θα συνέβαινε αν δύο πελάτες προσπαθούσαν να κλείσουν ταυτόχρονα το τελευταίο αεροπορικό εισιτήριο που διαθέτει μία εταιρεία. Θα πρέπει με κάποιο τρόπο να εξασφαλιστεί πως ο ένας από τους δύο θα δεσμεύσει τη θέση.

Δεν πρέπει ούτε να την πάρουν και οι δύο ούτε να μην την πάρει κανείς. Μια άλλη περίπτωση είναι αυτή των ΑΤΜ μιας τράπεζας. Έστω πως δύο άνθρωποι έχουν κοινό λογαριασμό σε μία τράπεζα. Και έστω πως πηγαίνουν και οι δύο την ίδια χρονική στιγμή να πάρουν χρήματα από δύο διαφορετικά ΑΤΜ. Τι θα συμβεί; Θα πάρουν τα χρήματα και οι δύο και αν ναι θα πάρουν τη σωστή απόδειξη για το υπόλοιπο του λογαριασμού τους και πως αυτό θα εξασφαλιστεί; Υπάρχει περίπτωση να συμβεί το εξής: Τη χρονική στιγμή t1 ο πρώτος από τους δύο συνδικαιούχους ζητά να κάνει ανάληψη. Το ΑΤΜ του ζητά το ποσό το οποίο θέλει να πάρει και διαβάζει το ποσό που πληκτρολογείται. Έστω για παράδειγμα ζητά 1000 ευρώ. Ο δεύτερος συνδικαιούχος τη χρονική στιγμή t2 ζητά να εισπράξει το ποσό των 2000 ευρώ. Η πρώτη διεργασία που συνδέται με τον πρώτο συνδικαιούχο αναζητά το υπόλοιπο του λογαριασμού. Η δεύτερη διεργασία που συνδέεται με τον δεύτερο συνδικαιούχο κάνει το ίδιο με κάποια καθυστέρηση. Η πρώτη διεργασία υπολογίζει το υπόλοιπο που απομένει στον λογαριασμό αν αφαιρεθεί το ποσό των 1000 ευρώ. Αν υποθέσουμε ότι ο λογαριασμός είχε 10000 ευρώ το υπόλοιπο είναι 9000 ευρώ. Και την ώρα που γίνεται αυτός ο υπολογισμός η δεύτερη διεργασία ξεκινά να υπολογίζει ακριβώς το ίδιο έχοντας ως υπόλοιπο το αρχικό υπόλοιπο που είναι οι δέκα χιλιάδες ευρώ και όχι αυτό που απομένει όταν ο πρώτος συνδικαιούχος πάρει το ποσό που ζήτησε. Έτσι το υπόλοιπο που τυπώνεται στον πρώτο πελάτη είναι 9000 ευρώ και στον δεύτερο 8000 ευρώ. Το σωστό θα ήταν να τυπωθεί στον δεύτερο πελάτη το ποσό των 7000 ευρώ. Για να μην συμβαίνουν τέτοια λάθη θα πρέπει τη στιγμή που ξεκινά να εκτελείται αυτό το τμήμα της διεργασίας το οποίο ονομάζεται κρίσιμο τμήμα να μην επιτρέπεται σε καμιά άλλη διεργασία να διακόψει την εκτέλεση του κρισίμου τμήματος της πρώτης. Το φαινόμενο αυτό δεν παρουσιάζεται μόνο με την ΚΜΕ αλλά και με την χρήση άλλων πόρων του συστήματος. Για παράδειγμα η χρήση του εκτυπωτή. Όταν μία διεργασία πρόκειται να εκτυπώσει ένα αρχείο αποθηκεύει το όνομα του αρχείου στην ουρά εκτύπωσης έως ότου ο εκτυπωτής ελευθερωθεί και μπορεί να εκτυπώσει. Μια διεργασία ελέγχει κάθε φορά αν υπάρχουν αρχεία προς εκτύπωση και αν ναι τότε τα τυπώνει και διαγράφει το όνομά τους από την ουρά εκτύπωσης. Υπάρχουν δύο μεταβλητές οι in και η out οι οποίες δείχνουν αντίστοιχα στην επόμενη ελεύθερη θέση στην ουρά και στη θέση του επόμενου αρχείου που θα εκτυπωθεί. Οι τιμές των δύο αυτών μεταβλητών είναι διαθέσιμες σε κάθε διεργασία. Υποθέτουμε ότι δύο διεργασίες αποφασίζουν να ζητήσουν εκτύπωση σχεδόν ταυτόχρονα. Τι θα συμβεί σε αυτή την περίπτωση; Έστω πως υπάρχουν κάποια αρχεία στην ουρά εκτύπωσης και η επόμενη θέση για την προσθήκη αρχείου είναι η θέση 10. Η πρώτη διεργασία (με ελάχιστη διαφορά από τη δεύτερη) αποφασίζει να εκτυπώσει και αποθηκεύει σε μία μεταβλητή της την τιμή 10 για να γνωρίζει σε ποιά θέση θα αποθηκεύσει το αρχείο που θέλει να εκτυπώσει. Ακριβώς μόλις αποθηκευτεί η τιμή αυτή γίνεται μία διακοπή στην εκτέλεση της διεργασίας γιατί ολοκληρώθηκε ο χρόνος που της είχε δωθεί. Συνεχίζει τώρα η εκτέλεση της δεύτερης διεργασίας η οποία επίσης θέλει να εκτυπώσει. Διαβάζει και εκείνη την τιμή της μεταβλητής σχετικά με την επόμενη ελεύθερη θέση για την προσθήκη ονόματος αρχείου για εκτύπωση και διαπιστώνει πως η τιμή αυτή είναι 10. Άρα αυτή τη στιγμή και οι δύο διεργασίες θεωρούν πως η επόμενη θέση της ουράς είναι η δέκατη. Ο χρόνος εκτέλεσης της δεύτερης διεργασίας δεν έχει ολοκληρωθεί και επομένως αυτή προσθέτει το όνομα του προς εκτύπωση αρχείου στη θέση 10. Η τιμή της μεταβλητής στην οποία αποθηκεύεται η επόμενη ελεύθερη θέση στην ουρά εκτύπωσης γίνεται 11. Η δεύτερη διεργασία συνεχίζει να εκτελείται. Κάποια στιγμή διακόπτεται η εκτέλεση για να δωθεί χρόνος στην επόμενη διεργασία. Όταν συνεχιστεί η εκτέλεση της πρώτης διεργασίας αυτή σκοπεύει να

αποθηκεύσει το όνομα του αρχείου που θέλει να εκτυπώση στη θέση 10 διότι αυτή την τιμή είχε κρατήσει πρν διακοπεί η εκτέλεσή της. Πηγαίνει λοιπόν στη θέση 10, διαγράφει το αρχείο της διεργασίας Β και γράφει το όνομα του δικού της αρχείου. Το αρχεό που είχε αποθηκευτεί στην ουρά εκτύπωσης από τη δεύτερη διεργασία δεν θα εκτυπωθεί ποτέ διότι έχει διαγραφεί. Αυτές οι καταστάσεις που προσπαθούν δύο ή περισσότερες διεργασίες να χρησιμοποιήσουν ταυτόχρονα πόρους του συστήματος συνθήκες ανταγωνισμού. Και για να μην συμβαίνουν ανεξήγητα λάθη αυτές οι συνθήκες πρέπει να μην δημιουργούνται. Ο τρόπος για να γίνει αυτό ονομάζεται αμοιβαίος αποκλεισμός. Όταν μία διεργασία μπαίνει στο κρίσιμο τμήμα της θα πρέπει το λειτουργικό σύστημα να εμποδίζει οποιαδήποτε άλλη να μπει στο δικό της κρίσιμο τμήμα. Αυτό ονομάζεται αμοιβαίος αποκλεισμός και θα δούμε στη συνέχεια πως γίνεται. Όταν μία διεργασία μπαίνει στο κρίσιμο τμήμα της και εκτελείται οποιαδήποτε προσπάθεια άλλης διεργασίας να μπει στο δικό της κρίσιμο τμήμα δεν επιτρέπεται. Σε αυτή την περίπτωση η δεύτερη εργασία περιμένει να εκτελεστεί. Όταν ολοκληρωθεί η εκτέλεση του κρίσιμου τμήματος της πρώτης διεργασίας τότε διακόπτεται η εκτέλεσή της (δεδομένου ότι ο χρόνος της έχει ήδη εκπνεύσει) και συνεχίζεται η εκτέλεση του κρίσιμου τμήματος της δεύτερης διεργασίας. Ο Dijikstra το 1965 [1] έθεσε και τους περιορισμούς για την παράλληλη εκτέλεση διεργασιών και τα κρίσιμα τμήματα:

> Δύο διεργασίες δεν επιτρέπεται να βρίσκονται ταυτόχρονα στο κρίσιμο τμήμα τους.
>
> Όταν μία διεργασία δεν βρίσκεται στο κρίσιμο τμήμα της δεν επιτρέπεται να εμποδίσει καμία διεργασία από την εκτέλεση του δικού της κρίσιμου τμήματος.
>
> Δεν επιτρέπεται μία διεργασία να περιμένει επ'αόριστο να μπει στο κρίσιμο τμήμα της ή όταν χρειστεί να μπουν ταυτόχρονα δύο διεργασίες στο κρίσιμο τμήμα τους να περιμένει η μία την άλλη και να μην εκτελείται καμιά από τις δύο.
>
> Δεν επιτρέπονται παραδοχές σε ό,τι αφορά την ταχύτητα ή το πλήθος των επεξεργαστών.
>
> Δεν επιτρέπεται κάποιες διεργασίες να μονοπωλούν το σύστημα. Πρέπει να υπάρχει κατα κάποιο τρόπο δικειοσύνη στην επιλογή της διεργασίας που θα μπει στο κρίσιμο τμήμα της.

Υπάρχουν διάφοροι τρόποι για να επιτευχθεί ο αμοιβαίος αποκλεισμός. Ένας από αυτούς είναι η απενεργοποίηση των διακοπών. Κάθε φορά που ολοκληρώνεται ο χρόνος που έχει στη διάθεσή της μία εργασία για να εκτελεστεί ενεργοποιείται μία διακοπή ρολογιού και η διεργασία αυτή εφόσσον δεν έχει ολοκληρωθεί η εκτέλεσή της μπαίνει στην ουρά αναμονής για να εκτελεστεί η επόμενη διεργασία της λίστας αναμονής. Θα μπορούσε μία διεργασία όταν μπει στο κρίσιμο τμήμα της να απανεργοποιεί αυτό το σύστημα διακοπών και να μην υπάρξει διακοπή εκτέλεσης και ενεργοποίηση νέας διεργασίας για όσο χρονικό διάστημα η εργασία αυτή εκτελεί το κρίσιμο τμήμα της. Η λύση αυτή έχει το μειονέκτημα πως η χρήση των διακοπών δίνεται στις διεργασίας και αυτό μπορεί να έχει απρόβλεπτες συνέπειες. Όπως για παράδειγμα να διακόψει μια διεργασία τις διακοπές του ρολογιού και να μην τις επαναφέρει.Υπήρξαν διάφορες ιδέες υλοποίησης του αμοιβαίου αποκλεισμού με πιο γνωστή του Ολλανδού μαθηματικού Dekker ο οποίος έδωσε λύση μέσω λογισμικού. Αλλά το 1981 ο Peterson έδωσε μια πιο απλή και αποτελεσματική λύση. Όταν μία διεργασία θέλει να μπει στο κρίσιμο τμήμα της και κατά συνέπεια να χρησιμοποιήσει

τις κοινόχρηστες μεταβλητές καλεί την enter_region με παράμετρο 0 ή 1. Με την κλήση αυτή η διεργασία θα περιμένει έως ότου γίνει εφικτό να μπει στο κρίσιμο τμήμα της. Όταν ολοκληρωθεί το κρίσιμο τμήμα της καλεί την leave_region προκειμένου να δηλώσει πως ολοκλήρωσε το κρίσιμο τμήμα της και μπορεί κάποια άλλη διεργασία που περιμένει να μπει στο δικό της κρίσιμο τμήμα να αποκτήσει πρόσβαση στις κοινόχρηστες μεταβλητές.

Πιο αναλυτικά διακρίνουμε δύο περιπτώσεις:

1. Οι διεργασίες ζητούν σε διαφορετική χρονική στιγμή να μπουν στο κρίσιμο τμήμα τους. Τι θα συμβεί;

   Τη στιγμή που κάποια διεργασία ζητήσει να μπει στο κρίσιμο τμήμα της και δεδομένου ότι υπάρχει αυτή η δυνατότητα καλείται η enter_region και το στοιχείο του πίνακα "interested" που αντιστοιχεί σε αυτή τη διεργασία (έστω η διεργασία 0) γίνεται true και η μεταβλητή turn γίνεται 0. Όταν κάποια άλλη διεργασία ζητήσει να μπει στο κρίσιμο τμήμα της θα περιμένει έως ότου η τιμή του στοιχείου 0 του πίνακα "interested" γίνει false. Αυτό θα συμβεί όταν η πρώτη διεργασία (δηλ. η διεργασία 0) καλέσει την leave_region.

2. Οι διεργασίες ζητούν την ίδια χρονική στιγμή να μπουν στο κρίσιμο τμήμα τους. Τι θα συμβεί;

   Υποθέτουμε πως και οι δύο διεργασίες ζητούν ταυτόχρονα να μπουν στο κρίσιμο τμήμα τους. Η μεταβλητή turn θα πάρει ως τιμή τον αριθμό της μιας διεργασίας και κατόπιν θα διαγραφεί αυτή η τιμή και θα πάρει την τιμή της επόμενης. Όταν φτάσουν και οι δύο στο while η πρώτη δεν θα το εκτελέσει και θα μπεί στο κρίσιμο τμήμα της γιατί η τιμή της turn είναι αυτή της δεύτερης διεργασίας ενώ η δεύτερη θα μπεί στο βρόχο και θα εκτελεί το while έως ότου η πρώτη διεργασία καλέσει την leave_region.

Ακολουθεί ο κώδικας του αμοιβαίου αποκλεισμού.

```
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;
}
```

Η λύση Peterson εκτελεί αναμονή με απασχόληση γιατί κάθε φορά που μία διεργασία θέλει να μπει στο κρίσιμο τμήμα της και δεν μπορεί εκτελεί ένα βρόχο while ο οποίος δεν κάνει τίποτα. Οι σηματοφορείς έρχονται να λύσουν αυτό το πρόβλημα.

13.5 Σηματοφορείς

Ο Dijkstra το 1965 [1] παρουσίασε τις λειτουργίες P και V. Ένας σηματοφορέας ορίζεται σαν μια ακέραιη μεταβλητή. Οι λειτουργίες P και V ορίζονται ως εξής:

P(s): while s<= 0 do skip;

s:=s-1

V(s): s:=s+1

Υποθέτουμε πως δύο διεργασίες δεν θα ζητήσουν να εκτελέσουν ταυτόχρονα τις λειτουργίες P ή V. Αν κάτι τέτοιο προκύψει τότε τυχαία καθορίζεται ποια διεργασία θα έκτελέσει πρώτη τη λειτουργία που ζητά. Όταν έχουμε n διεργασίες το πρόβλημα του κρίσιμου τμήματος λύνεται ως εξής:

mutex:=1;

parbegin

P1: repeat P(mutex); KT1; V(mutex); μη KT1; Until false ||

.

.

.

Pi: repeat P(mutex); KTi; V(mutex); μη KTi; Until false ||

.

.

.

Pn: repeat P(mutex); KTn; V(mutex); μη KTn; Until false ||

parend

όπου mutex είναι η ακέραια μεταβλητή η οποία αποτελεί τον σηματοφορέα. Η τιμή της μεταβλητής – σηματοφορέα είναι ίση με το μηδέν κάθε φορά που κάποια διεργασία μπαίνει στο κρίσιμο τμήμα της. Η τιμή αυτή γίνεται ένα όταν βγαίνει από το κρίσιμο τμήμα. Μία μόνο διεργασία μπορεί να αλλάξει την τιμή του σηματοφορέα και επομένως μόνο μία διεργασία μπαίνει στο κρίσιμο τμήμα της.

13.6 Δίκτυα

13.6.1 Κατηγορίες Δικτύων

Αν και δεν υπάρχει σαφής τρόπος διαχωρισμού των δικτύων υπάρχουν κάποιες βασικές ιδιότητες που μας επιτρέπουν να διακρίνουμε τα δίκτυα σε κατηγορίες. Οι πιο γνωστές εκ των οποίων είναι η τεχνολογία μετάδοσης, το μέσο διασύνδεσης και η άλλη η γεωγραφική κάλυψη (κλίμακα).

Ως προς την πρώτη ιδιότητα, αυτή της τεχνολογίας μετάδοσης, διακρίνουμε τα δίκτυα σε δίκτυα εκπομπής (broadcast networks) και δίκτυα από σημείο σε σημείο (point to point).

Δίκτυα εκπομπής: όλες οι μηχανές του δικτύου μοιράζονται το ίδιο κανάλι επικοινωνίας. Η επικοινωνία γίνεται με την αποστολή πακέτου από κάποια μηχανή σε όλες τις υπόλοιπες. Για να παραληφθεί το πακέτο από τη σωστή μηχανή υπάρχει ένα πεδίο διεύθυνσης στο ίδιο το πακέτο στο οποίο καθορίζεται ο παραλήπτης. Ο παραλήπτης επεξεργάζεται αυτό το οποίο λαμβάνει με το πακέτο ενώ οι υπόλοιπες μηχανές απλά το αγνοούν. Υπάρχει επίσης η δυνατότητα να σταλεί ένα πακέτο προς όλους τους παραλήπτες χρησιμοποιώντας κάποια συγκεκριμένη τιμή στο πεδίο διεύθυνσης. Αυτός ο τρόπος ονομάζεται ευρείας μετάδοσης. Τέλος σε κάποια δίκτυα υπάρχει η δυνατότητα να σταλεί ένα πακέτο σε ένα υποσύνολο των μηχανών του δικτύου. Στην περίπτωση αυτή στο πεδίο της διεύθυνσης φαίνεται ότι πρόκειται για πολυδιανομή καθώς και η ομάδα (το υποσύνολο των μηχανών) την οποία αφορά το πακέτο. Όταν το πακέτο απευθύνεται σε συγκεκριμένη ομάδα το επεξεργάζονται όλες οι μηχανές που ανήκουν στην ομάδα.

Δίκτυα από σημείο σε σημείο: Στην κατηγορία αυτή ανήκουν τα δίκτυα στα οποία οι μηχανές συνδέονται με πολλές συνδέσεις μεταξύ τους. Και επειδή πολλές φορές συμβαίνει ένα πακέτο για να φτάσει στον προορισμό του να χρειαστεί να περάσει από πολλές ενδιάμεσες μηχανές ένα σημαντικό θέμα είναι να βρεθεί το καλύτερο δυνατό δρομολόγιο μεταξύ των μηχανών.

Ως προς την δεύτερη ιδιότητα, αυτή της τεχνολογίας μετάδοσης, τα δίκτυα διακρίνονται σε ενσύρματα και ασύρματα.

Ενσύρματο χαρακτηρίζεται ένα δίκτυο υπολογιστών στο οποίο η διασύνδεση των μηχανών για λόγους επικοινωνίας επιτυγχάνεται με τη χρησιμοποίηση κάποιου τύπου καλωδίου. Ασύρματο δίκτυο χαρακτηρίζεται το τηλεπικοινωνιακό δίκτυο το οποίο χρησιμοποιεί, ραδιοκύματα ως φορείς πληροφορίας. Τα δεδομένα μεταφέρονται μέσω ηλεκτρομαγνητικών κυμάτων και η συχνότητα εξαρτάται κάθε φορά από τον ρυθμό μετάδοσης δεδομένων που απαιτείται να υποστηρίζει το δίκτυο. Η ασύρματη επικοινωνία, σε αντίθεση με την ενσύρματη, δεν χρησιμοποιεί ως μέσο μετάδοσης κάποιον τύπο καλωδίου.

Ως προς την γεωγραφική κάλυψη τα δίκτυα χωρίζονται σε τοπικά, μητροπολιτικά, ευρείας περιοχής και διαδίκτυα.

Τα τοπικά δίκτυα τα οποία συνήθως αποκαλούνται LAN (Local Area Networks) είναι δίκτυα που συνδέουν υπολογιστές που βρίσκονται σε κοντινές μεταξύ τους αποστάσεις. Είναι αυτά που συνήθως χρησιμοποιούνται σε εταιρείες, πανεπιστήμια, εργοστάσια.

Τα μητροπολιτικά δίκτυα τα οποία συνήθως αποκαλούνται MAN (Metropolitan Area Network) καλύπτει μεγαλύτερες αποστάσεις όπως για παράδειγμα μία πόλη. Το πιο γνωστό παράδειγμα σε αυτή την κατηγορία είναι το δίκτυο καλωδιακής τηλεόρασης που μπορεί να υπάρχει σε μία πόλη.

Τα δίκτυα ευρείας περιοχής τα οποία συνήθως αποκαλούνται WAN (Wide Area Network) καλύπτουν μεγάλες γεωγραφικές περιοχές όπως μία χώρα ή μία ήπειρο και μπορούν να συνδέσουν ακόμη και περισσότερα από ένα τοπικά δίκτυα καθώς και ομάδες τοπικών δικτύων. Τα διαδίκτυα είναι ένα σύνολο διασυνδεδεμένων δικτύων. Είναι δίκτυα ευρείας περιοχής και καλύπτουν γεωγραφικές περιοχές μιάς ή περισσοτέρων ηπείρων διασυνδέοντας επιμέρους δίκτυα. Σε ένα διαδίκτυο μπορεί να συνυπάρχουν διασυνδεδεμένοι υπολογιστές και δίκτυα υπολογιστών που χρησιμοποιούν διαφορετικές τεχνολογίες και διαφορετικά λειτουργικά συστήματα. Το ”Διαδίκτυο” (Internet) είναι το μεγαλύτερο τέτοιου είδους δίκτυο.

13.6.2 Ιεραρχίες Πρωτοκόλλων

Τα δίκτυα οργανώνονται σε μια στοίβα επιπέδων στην οποία το ένα επίπεδο χτίζεται πάνω στο προηγούμενο. Κάθε επίπεδο προσφέρει υπηρεσίες στο ανώτερό του επίπεδο αποκρύπτοντας μέρος της πληροφορίας που αφορά στις λεπτομέρειες υλοποίησης της των υπηρεσιών που προσφέρει το κατώτερο στο ανώτερο επίπεδο. Ο τρόπος που πραγματοποιείται μία συνομιλία μεταξύ δύο μηχανών στο επίπεδο n ονομάζεται πρωτόκολλο επιπέδου n. Το πρωτόκολλο ουσιαστικά καθορίζει τους κανόνες με τους οποίους γίνεται η επικοινωνία. Δύο ευρέως γνωστές αρχιτεκτονικές δικτύων είναι το μοντέλο OSI και το μοντέλο TCP/IP.
Το μοντέλο OSI (Open Systems Interconnection) έχει 7 επίπεδα ιεραρχίας:

Φυσικό επίπεδο
Είναι το κατώτερο επίπεδο και ασχολείται με τη μετάδοση των bits στο κανάλι επικοινωνίας.

Επίπεδο Διασύνδεσης Δεδομένων
Ελέγχει τη ροή των δεδομένων και την αξιόπιστη μετάβαση των πακέτων. Παίρνει τα δεδομένα από το φυσικό επίπεδο και να τα προωθεί στο επίπεδο δικτύου αφού ανιχνεύσει και διορθώσει σφάλματα μετάδοσης.

Επίπεδο Δικτύου
Ελέγχει τη λειτουργία του υποδικτύου. Αν πολλά πακέτα βρίσκονται στο διαδίκτυο την ίδια χρονική στιγμή παρεμποδίζει το ένα το άλλο. Το επίπεδο δικτύου αναλαμβάνει τον έλεγχο της συμφόρησης. Επίσης αν το πακέτο πηγαίνει από ένα δίκτυο σε ένα άλλο μπορούν να υπάρξουν προβλήματα. Για παράδειγμα να διαφέρουν στην διευθυνσιοδότηση,

να μη γίνει αποδεκτό το πακέτο από το δίκτυο προορισμού λόγω μεγέθους, να διαφέρουν τα πρωτόκολλα.

Επίπεδο μεταφοράς
Δέχεται δεδομένα από το επίπεδο συνδιάλεξης (το ανώτερό του επίπεδο), τα διασπά αν χρειάζεται και τα μεταβιβάζει στο επίπεδο δικτύου.

Επίπεδο Συνδιάλεξης
Πραγματοποιεί τις κατάλληλες λειτουργίες ώστε διαφορετικές μηχανές να συνδιαλέγονται χωρίς προβλήματα μεταξύ τους μεσω του δικτύου. Σε περίπτωση προβλήματος στη σύνδεση, φροντίζει για την αποκατάστασή του.

Επίπεδο Παρουσίασης
Ασχολείται με τις κατάλληλες ενέργειες ώστε τα δεδομένα που μεταδίδονται να αναπαριστώνται με συμβατό τρόπο μεταξύ των διαφόρων υπολογιστών. Αυτό επιτυγχάνεται για παράδειγμα με τον ορισμό δομών δεδομένων με αφαιρετικό τρόπο.

Επίπεδο Εφαρμογής
Αποτελεί το ανώτερο επίπεδο. Περιέχει μια ποικιλία πρωτοκόλλων με εφαρμογές που εξυπηρετούν τον χρήστη. Τέτοια πρωτόκολλα είναι το πρωτόκολλο μεταφοράς υπερκειμένου, το ηλεκτρονικό ταχυδρομείο κ.λπ.

13.7 Διαδραστικό Υλικό – Σύνδεσμοι

- https://en.wikipedia.org/wiki/Operating_system
- https://en.wikipedia.org/wiki/Computer_network
- https://en.wikipedia.org/wiki/Process_management_%28computing%29
- https://el.wikipedia.org/wiki/%CE%9C%CE%BF%CE%BD%CF%84%CE%AD%CE%BB%CE%BF_%CE%B1%CE%BD%CE%B1%CF%86%CE%BF%CF%81%CE%AC%CF%82_OSI
- https://en.wikipedia.org/wiki/Internet_protocol_suite

13.8 Ασκήσεις

1. Τι είναι κρίσιμο τμήμα μιας διεργασίας και πως υλοποιείται ο αμοιβαίος αποκλεισμός.
2. Ποιά η διαφορά προγράμματος και διεργασίας και σε ποιές καταστάσεις μπορεί να βρίσκεται μία διεργασία;
3. Τι είναι πολυπρογραμματισμός;

4. Υποθέτουμε ότι 3 διεργασίες (δ1,δ2,δ3) με χρόνο εκτέλεσης 10, 12, 16 microsecs θέλουν να χρησιμοποιήσουν την ΚΜΕ. Η σειρά άφιξης είναι πρώτα η δ1 μετά η δ2 και τελευταία η δ3. Σε πόσο χρόνο θα ολοκληρωθεί κάθε μία από τις τρεις αν η ΚΜΕ εκτελεί για 4 microsecs μια διεργασία και στη συνέχεια εκτελεί την επόμενη. Αρχικά υποθέτουμε η σειρά άφιξης είναι και η σειρά εισόδου στην ΚΜΕ και κάθε φορά που τελειώνει ο χρόνος για μία διεργασία και ολοκληρωθεί μπαίνει τελευταία στην ουρά αναμονής. Αν υποθέσουμε πως η ΚΜΕ εκτελεί κάθε φορά την μικρότερη σε διάρκεια εκτέλεσης διεργασία και ότι όλες οι διεργασίες βρίσκονται ταυτόχρονα στην ουρά αναμονής πόσος χρόνος θα απαιτηθεί για κάθεμιά από τις δ1, δ2, δ3;

5. Υποθέτουμε πως έχουμε έναν επεξεργαστή και εκτελεί προγράμματα που κατά μέσο όρο απαιτούν 2 secs πρόσβαση στην ΚΜΕ και 10 secs στις μονάδες εισόδου εξόδου. Ποιό είναι το ποσοστό του χρόνου που η ΚΜΕ μένει ανενεργή;

Βιβλιογραφία

[1] Al Aho and Jeff Ullman: ”Foundations of Computer Science”, W.H.Freeman, 1992, free online.

[2] E. Dijkstra “Concurrent Programming, Mutual Exclusion” 1965.

[3] Andrew S. Tanenbaum “Modern Operating Systems, third edition”

[4] Γ.Κ. Παπακωνσταντίνου, Ν.Α Μπιλάλης, Π.Δ. Τσανάκας “Λετουργικά Συστήματα”

[5] Andrew S. Tanenbaum “ Computer Networks, fourth edtion”

[6] Behrouz Forouzan “Εισαγωγή στην Επιστήμη των Υπολογιστών” τρίτη έκδοση 2014