**Χρονοδρομολόγηση FCFS (First-Come, First-Served) 1.** Απλός, εύκολος και δίκαιος στην υλοποίηση. **2.** Δίνει προτεραιότητα στις διεργασίες που έφτασαν πρώτες. **3.** Δεν λαμβάνει υπόψη τον χρόνο εκτέλεσης μιας διεργασίας, οπότε μπορεί να προκαλέσει μεγάλους χρόνους αναμονής για πολύ υπολογιστικά απαιτητικές διεργασίες -> για να το μειώσω **ShortestJobFirst 4.** Δεν έχει ανάγκη σε υλικό

**Χρονοδρομολόγηση RR (Round Robin)1.** Δίκαιη κατανομή του χρόνου επεξεργασίας σε όλες τις διεργασίες με ορισμένο κβάντο χρόνου (μέσω καταχωρητή μετρητή) **2.** Κυκλική πρόσβαση στη CPU 3. Προσωπικός υπολογιστής: ψευδαίσθηση παράλληλης εκτέλεσης πολλαπλών διεργασιών και χαμηλός χρόνος αποκρισιμότητας 4. Υπάρχει overhead λόγω της συχνής εναλλαγής διεργασιών 5. ο χρονοδρομολογητής πρέπει να εκτελείται συχνά context-switch -> αποθήκευση κάθε φορά σε καταχωρητές της κατάστασης κάθε διεργασίας για να συνεχιστεί στη συνέχεια 6. Μπορεί να προκαλέσει αυξημένο χρόνο αναμονής για μακροχρόνιες διεργασίες 🡪 εκχωρείται ίδιο χρονικό κβάντο σε όλες όποτε πιθανόν να περιμένουν για αρκετές εκτελέσεις πρωτού τους δοθεί πάλι η σειρά να εκτελεστούν

**Χρονοδρομολόγηση ShortestJobFirst:** 1. επιλέγεται η διεργασία με το μικρότερο χρόνο burst CPU (βέλτιστος) 2. μη υλοποιήσιμος αφού γενικά το μήκος επόμενου burst δεν ειναι γνωστό -> προσέγγιση με βάση προηγούμενες τιμές – ξεκινάω με έναν random χρόνο εκτέλεσης 3. φαινόμενο λιμοκτονίας --> χρησιμοποιούμε Γήρανση (aging) με την πάροδο του χρόνου αυξάνεται η προτεραιότητα των διεργασιών που δεν εκτελούνται

Ένας non-preemptive χρονοδρομολογητής: 1. εκτελείται όταν μια διεργασία μεταβεί σε κατάσταση αναμονής ή όταν τερματίσει. 2. Απλός - Μικρότερη πολυπλοκότητα 3. Οι εκτελούσες διεργασίες αποφασίζουν πότε θα προχωρήσουν την εκτέλεση σε άλλες διεργασίες 4. λαμβάνει απόφαση χρονοδρομολόγησης μόνο στις μεταβάσεις: RUNNING 🡪WAITING, RUNNING 🡪 TERMINATED Που αντιστοιχούν σε διακοπές λογισμικού: η τρέχουσα διεργασία προκαλεί trap για κλήση συστήματος ή pagefault 5. Ανεπάρκεια σεναρίων που απαιτούν δίκαιη και ισότιμη κατανομή του χρόνου CPU Εάν μια διεργασία δεν παραχωρήσει εθελοντικά τον έλεγχο, άλλες διεργασίες μπορεί να μην έχουν ποτέ την ευκαιρία να εκτελεστούν. ανεπίτρεπτα μεγάλοι χρόνοι αναμονής για ορισμένες διεργασίες 6. κατάλληλος για υπερυπολογιστές

Ένας preemptive χρονοδρομολογητής: 1. μπορεί να διακόπτει τις διεργασίες 2. λαμβάνει απόφαση χρονοδρομολόγησης στις μεταβάσεις: RUNNING 🡪WAITING, RUNNING 🡪 TERMINATED, RUNNING 🡪READY, WAITING 🡪 READY Που αντιστοιχούν σε διακοπές υλικού από εξωτερικούς παράγοντες: ο χρονιστής διακόπτει τη CPU γιατί το κβάντο χρόνου τελέιωσε ή ο δίσκος διακόπτει τη CPU γιατι μία μεταφορά ολοκληρώθηκε 3. μπορούν να οδηγήσουν σε αλλαγή της τρέχουσας διεργασίας και με force 4. προσφέρει τη δυνατότητα προτεραιότητας και ελέγχου διεργασιών

αποτελεσματική διαχείριση του χρόνου CPU 5. καλύτερη αποκρισιμότητα 6. διασφάλιση ότι κάθε διεργασία θα έχει ευκαιρία για εκτέλεση 7. πρόσθετη πολυπλοκότητα 🡪 διαχείριση διακοπτών/προτεραιοτήτων 8. προσωπικός υπολογιστής

Χάρτης Μνήμης 1. δημιουργείται με τη γέννηση της διεργασίας 2. βρίσκεται στο PCB της 3. επικοινωνεί με το λειτουργικό και μας ενημερώνει κάθε φορά αν περιλαμβάνει όλη τη πληροφορία για τη μνήμη κάθε διεργασίας-> πόσες σελίδες έχει και που βρίσκονται - μία σελίδα μπορεί να βρίσκεται στη μνήμη, στο δίσκο ή να μην έχεις το δικαίωμα να τη φορτώσεις 4. τον συμβουλεύεται το ΛΣ αν γίνει page fault 5. ανανεώνεται αν το πρόγραμμα ζητήσει - ζητήσει να μεγαλώσει τη στοίβα του - ελευθερώσει δυναμικά μνήμη στο σωρό (malloc/free) - απεικονίσει ένα αρχείο στη μνήμη

Πίνακας Σελίδων - γράφεται από το ΛΣ- συμβουλεύεται από αυτό το υλικό(MMU) για αντιστοίχιση - περιλαμβάνει μεταφράσεις των λογικών διευθύνσεων σε φυσικές της μνήμης - περιλαμβάνει:1. αριθμό σελίδας 2. bit πρόσβασης (r,w,x)3. bit εγκυρότητας (valid/invalid) -> invalid -> τότε δε βρίσκεται στην κύρια μνήμη αλλά στο δίσκο και πρέπει να γίνει μεταφορά -> page fault -> η διεργασία πέφτει σε sleep mode και γίνεται context switch με την επόμενη διεργασία του χρονοδρομολογητή στην ουρά αναμονής επειδή η μεταφορά από το δίσκο είναι χρονοβόρα διαδικασία -> χάρτης μνήμης -> αλλαγή page table με φυσική μνήμη-> invalid -> valid 4. modify bit - αν το block έχει αλλαχθεί από τότε που μεταφέρθηκε από το δίσκο οπότε αν όχι τότε τίποτα αν ναι τότε θα πρέπει να γίνει replace στο δίσκο με νέο αντίγραφο

Εσωτερικός κατακερματισμός- ύπαρξη αχρησιμοποίητων περιοχών μνήμης μέσα σε περιοχές δεσμευμένες από διεργασίες - κάθε διεργασία καταλαμβάνει περισσότερο χώρο από ό,τι χρειάζεται γιατί η μνήμη της μπορεί να μην είναι πολλαπλάσιο του μεγέθους της σελίδας

Ένα softlink λειτουργεί σαν ένας pointer που δείχνει στο όνομα ενός αρχείου (σαν ένα ζεύγος ονομάτων -> ένα όνομα που δείχνει στο όνομα του αρχείου) Το hardlink λειτουργεί σαν ένας pointer που δείχνει στο ίδιο το αρχείο (στο ζεύγος όνομα αρχείου και inode (FCB) Το ΛΣ αποθηκεύει τον αριθμό hardlinks/softlinks που δείχνουν σε ένα αρχείο (reference counter) για αυτό: Αν διαγραφτούν όλα τα hardlinks ενός αρχείου τότε διαγράφεται και το ίδιο το αρχείο Αν διαγραφτούν όλα τα softlinks τότε ΤΙΠΟΤΑ Αν διαγραφτεί το αρχείο τότε softlinks -> NULL (εξακολουθούν να υπάρχουν)

Open() -> Το ΛΣ βρίσκει το current directory από το PCB της τρέχουσας διεργασίας - Διαβάζει το inode id του αρχείου που θέλουμε - Βρίσκει το inode id στο ευρετήριο και παίρνουμε το FCB του αρχείου, δηλαδή όλα τα μεταδεδομένα του - Βλέπουμε αν έχουμε δικαιώματα πρόσβασης - Ενημερώνουμε της δομες του ΛΣ (global – local πινακας ανοιχτών αρχείων) - Επιστροφή file descriptor (δηλαδή επιστρέφουμε τον αριθμό θέσης στο local table) Read() - Από το file descriptor βρίσκω τη θέση του αρχείου στο local πίνακα - Βρίσκω τη θέση του στον global - Παίρνω το FCB του - Ανάγνωση δεδομένων (κάνω reαd) - Ενημερώνω τον δείκτη του read στο local πίνακα για τη διεργασία

Σελιδοποίηση - διαχωρισμός της διαθέσιμης μνήμης σε blocks/σελίδες - κάθε σελίδα = page number + offset (offset είναι η μετατόπιση στη σελίδα page\_number[offset]) - κάθε διεργασία μπορεί να καταλαμβάνει όσο διαθέσιμα pages χρειάζεται - Λύνει τον εξωτερικό κατακερματισμό - Μειονέκτημα: εσωτερικός κατακερματισμός - Το μέγεθος της σελίδας συνηθίζεται να είναι δύναμη του 2 για να πραγματοποιείται εύκολα ο διαχωρισμός αριθμού σελίδας (pagenumber) και μετατόπησης (offset).

Εξωτερικός κατακερματισμός 1. ύπαρξη πολλών μικρών αδέσμευτων περιοχών μνήμης 2. δεν μπορεί να εξυπηρετηθεί αίτηση για δέσμευση μεγάλου ποσού συνεχόμενης μνήμης 3. παρόλο που συνολικά ο διαθέσιμος χώρος επαρκεί

Συνεχόμενη ανάθεση Κύριας Μνήμης - Το ΛΣ παραμένει συνεχώς φορτωμένο σε ένα εύρος διευθύνσεων - Κάθε νέα διεργασία έχει συγκεκριμένες απαιτήσεις σε μνήμη - Οι διεργασίες περιμένουν μέχρι κατάλληλη συνεχόμενη διαμέριση να γίνει διαθέσιμη -> αναμονή - Πρόβλημα: εξωτερικός κατακερματισμός -> επίλυση σύμπτυξη (υψηλό κόστος σε χρόνο και υλικό) - Απαιτούνται δύο επιπλέον καταχωρητές που ελέγχονται από την MMU και προσδιορίζουν τα επιτρεπτά όρια προσπέλασης μνήμης (Base register- Limit register )

Για να υλοποιηθεί ένα σύστημα αρχείων το ΛΣ χρειάζεται 1. Ένα μέσο αποθήκευσης π.χ. δίσκος 2. Σχεδιασμός και υλοποίηση αποθήκευσης δεδομένων σε αυτό(αρχείων και καταλόγων) -> Πραγματικά δεδομένα αρχείων/καταλόγων - Μεταδεδομένα αρχείων - Ευρετήριο αρχείων και καταλόγων (Είναι ένας γρήγορος τρόπος να οργανώνονται τα αρχεία με βάση το inode id για να βρίσκονται γρήγορα (στο δίσκο)) 2. Δομές δεδομένων και κώδικας (κλήσεις συστήματος) για το χειρισμό των αρχείων και καταλόγων -> Ανοιχτά αρχεία με το FCB τους - Global (όλα τα FCBs) : Ανοιχτά αρχεία ανά διεργασία, τα δικαιώματα τους - Local πίνακας στο PCB κάθε διεργασίας με τα ανοιχτά αρχεία-> Για κάθε αρχείο έχει: 1. Έναν δείκτη στον global πίνακα αρχείων 2. δικαιώματα 3. έναν δείκτη που κάνει read, έναν άλλον για write ανάλογα με τα δικαιώματα του αρχείου -> έτσι όταν καλώ π.χ. τη read μέσα σε while να μην διαβάζει το file κάθε φορά από την αρχή αλλά να το διαβάζει ολόκληρο προχωρώντας κάθε φορά κατά ένα bit αποθηκεύοντας τη θέση του στο δείκτη -> fd 4. Τρέχουσα θέση ανά αρχείο – fd (file descriptor)

Μονοπάτι (path) -> ακολουθία αρχείων -> Όποιο αρχείο ξεκινάει με ‘.’ είναι κρυφό Σχετικό path 1. Δε ξεκινάει από το root -> Για να βρούμε το file το ΛΣ βρίσκει το current directory από το PCB της τρέχουσας διεργασίας και διασχίζει το path ανοίγοντας τα αντίστοιχα directories Απόλυτο path (από το root) -> Για να βρούμε το file πάμε στο root και διασχίζουμε το path ανοίγοντας τα αντίστοιχα directories Τι συμβαίνει όταν μεταφέρουμε ένα αρχείο σε άλλο directory 1. Σβήνω την καταχώρηση στο παλιό directory 2. Προσθέτω τη καινούργια καταχώρηση στο νέο directory 3. Δε γίνονται αλλαγές στο μέσο αποθήκευσης (δίσκος) Δημιουργία αρχείου 1. Δέσμευση χώρου για τα δεδομένα του και το FCB της 2. Ενημέρωση όλων των δομών του ΛΣ π.χ. ευρετήριο, local πίνακας διεργασίας ανοιχτών αρχείων και global πίνακας ανοιχτών αρχείων

Από την οπτική του χρήστη ένα σύστημα αρχείων προσφέρει: 1. Δυνατότητα πρόσβασης σε αρχεία 2. Οργάνωση των αρχείων σε καταλόγους 3. Δικαιώματα πρόσβαση 4. Κλήσεις συστήματος για το χειρισμό αρχείων και καταλόγων

Κάθε αρχείο έχει τη δικιά του μοναδική ταυτότητα η οποία αποθηκεύεται σε μία δομή που ονομάζεται FCB(inode) και περιέχει όλα τα μεταδεδομένα του δηλαδή: 1. Μέγεθος αρχείου 2. Δικαιώματα πρόσβασης 3. Χρήστης, λίστες πρόσβασης 4. Τοποθεσία δεδομένων 5. Ημερομηνίες -> Κάθε FCB προσδιορίζεται από ένα inode id μοναδικό -> Σε ένα directory έχουμε ζεύγη ονομα αρχείου – inode id

Οι παρακολουθητές είναι κατασκευές των γλωσσών προγραμματισμού που ενσωματώνουν μεταβλητές, διαδικασίες πρόσβασης και κώδικα αρχικοποίησης σε έναν αφηρημένο τύπο δεδομένων. Μόνο μια διεργασία μπορεί να είναι ενεργή στον παρακολουθητή κάθε φορά (οι υπόλοιπες τοποθετούνται σε ουρά αναμονής ) , αφού η πρόσβαση γίνεται με χρήση των δικών του διαδικασιών πρόσβασης (μεταβλητές συνθήκης cwait , csignal). Με αυτό τον τρόπο υλοποιείται αυτόματα ο αμοιβαίος αποκλεισμός , σε σχέση με άλλα κλειδώματα που πρέπει να υλοποιηθεί συγκεκριμένα για κάθε διεργασία , και επιπλέον η κατασκευή του παρακολουθητή είναι ευκολότερη.

Συγχρονισμός Πρόβλημα παραγωγός-καταναλωτής -> λύνεται με 3 mutexes (space, items, mutex) Κρίσιμο τμήμα 1. επιτρέπει μόνο μία διεργασία κάθε φορά 2. Προστατευόμενο τμήμα/ εκτελείται ατομικά lock/unlock -> Λύση Peterson -> δε μπορεί να χρησιμοποιηθεί γιατί υλοποιείται μόνο για 2 διεργασίες και ο CPU κάθε φορά μπορεί να αναδιατάσσει τις εντολές του προγράμματος και αυτές να μην εκτελούνται με τη σωστή σειρά 2. ατομικές εντολέ; -> λειτουργούν ατομικά και εκτελούνται από το υλικό 3. Spinlocks -> ζητούν συνεχώς να μπουν στο κρίσιμο τμήμα μέχρι να το εκτελέσουν (ενεργός αναμονή)-> χρησιμοποιούνται σε προγράμματα με πολύ μικρούς χρόνους αναμονής -> σπαταλούν κύκλους CPU χωρίς να απαιτούν τη συνδρομή του ΛΣ 4. Mutexes-> μία διεργασίας καλεί να μπει στο ΚΤ -> αναστέλλεται μέχρι να ελευθερωθεί (waiting) -> ο επεξεργαστής ελευθερώνεται -> ο χρονοδρομολογητής επιλέγει την επόμενη διεργασία από το ΛΣ -> η διεργασία αποκτά state: READY 5. semaphores -> ένας ακέραιος αριθμός -> αρχικοποίηση του (αν =1 τότε ισοδυναμούν με mutex)-> όταν μία διεργασία προσπαθήσει να μπει στο ΚΤ τότε κάνει wait (semaphores -1)-> αν semaphores >=0 τότε η διεργασία μπαίνει στο ΚΤ -> αλλιώς συνεχίζει στο wait -> όταν η διεργασία του ΚΤ τελειώνει τότε καλείται η signal (semaphoresn+1) -> αν semaphores >=0 τότε μπαίνει η επόμενη της ουράς αναμονής |||| τα semaphores ελέγχουν την πρόσβαση σε περισσότερα από ένα νήματα- διεργασίες και γίνεται αμοιβαίος αποκλεισμός. |||||| οι εντολές signal() and wait() είναι ατομικές εντολές δηλαδή: εκτελούνται από το υλικό και έχουν αποκλειστικότητα (δηλ. κάνουν lock το ΚΤ)

Από 1 -> Ν διεργασίες : 1. Η διεργασια μπορει να ξυπνησε αλλες διεργασιες 2. Μπορει ο χρηστης να ανοιξε αλλες εφαρμογές που εφεραν αλλες Ν διεργασιες 3. Μπορει να υπήρχαν σε sleep mode (wait) και το τέλος αυτής της διεργασίας να τις ξύπνησε

Πολυεπίπεδη σελιδοποίηση: Πλεονέκτημα: χρειάζεται μικρότερος χρόνος για την διατήρηση των πινάκων σελίδων - Μειονέκτημα: αύξηση του κόστους διάσχισης των πινάκων σελίδων (π.χ. σε περίπτωση TLB miss).

Μία διεργασία θέλει να ενημερώσει μία άλλη διεργασία για την τιμή μίας μεταβλητής. Κοινή μνήμη (όπως fork()) + Pipes

Αναφέρετε τι πληροφορία κρατάει ο Πίνακας ελέγχου Διεργασίας(PCB) που να υποστηρίζει: Γενικά περιέχει: 1. Ταυτότητα διεργασίας (PID) 2. State 3. PC – registers 4. CPU scheduling info (προτεραιότητες) ->FCFS, SJF, RR, Priority Scheduling, Kριτήρια αξιολόγησης: throughput, response time, turnaround time(ολοκλήρωσης), waiting time, CPU utilization(χρησιμοποίηση) 5. Memory management info (δεσμευμένες περιοχές μνήμης, είδος, όρια πρόσβασης) 6. Accounting info (time, CPU usage) 7. Πληροφορίες για τον έλεγχο πρόσβασης(ταυτότητα χρήστη, δικαιώματα πρόσβασης, δυνατότητες) 8. Current directory 9. I/O status info (πίνακας ανοιχτών αρχείων)

τη μετάφραση των λογικών διευθύνσεων σε φυσικές -> χάρτης μνήμης

τη πρόσβαση στο Σύστημα Αρχείων -> current directory και Local πίνακας Ανοικτών αρχείων: Μια δομή δεδομένων που καταγράφει τα ανοιχτά αρχεία της διεργασίας. Κάθε ανοιχτό αρχείο έχει αντιστοιχιστεί ένας κατάλογος αρχείου που περιέχει πληροφορίες όπως η θέση του δείκτη αρχείου και τα δικαιώματα πρόσβασης.

την εναλλαγή περιβάλλοντος (context-switch) *[είναι η διαδικασία στην οποία το ΛΣ αλλάζει τη διεργασία που εκτελείται στη CPU, αποθηκεύει τη κατάσταση της τρέχουσας διεργασίας στο PCB της ή την επαναφέρει]*-> στιγμιότυπο της αρχιτεκτονική κατάσταση της τρέχουσας διεργασίας δηλαδή αντίγραφα των τιμών όλων των καταχωρητών της για να μπορεί να την επαναφέρει

*Η εναλλαγή περιβάλλοντος ανάμεσα σε νήματα της ίδιας διεργασίας είναι φθηνότερη, γιατί όλα τα νήματα τρέχουν στο ίδιο περιβάλλον μνήμης 🡪 δε χρειάζεται αλλαγή της κατάστασης της MMU*

Αναφερεται τουλάχιστον 5 λογους για τους οποίους μία διεργασία ΔΕΝ ολοκλήρωσε το κβάντο χρόνου της : 1. Ολοκληρώθηκε νωρίτερα 2. Στάλθηκε signal που την έκανε kill από το πρόγραμμα 3. Μπορεί να κάνει wait μέχρι να διαβάσει κάτι 4. Segmentation fault -> διαίρεση με το 0/ null pointer dereference 5. Μπορεί να ολοκληρώθηκε η μεταφορά διεύθυνσης μνήμης από το δίσκο και χρονοδρομολογητής να την έκανε wait 6. Permission error -> να προσπάθησε να διαβάσει κάτι που δεν έχει άδεια

Το ΛΣ εμπλέκεται στις παρακάτω περιπτώσεις:

1. το πρόγραμμα του χρήστη καλεί μία συνάρτηση/υπορουτίνα
   1. σε περίπτωση που η συνάρτηση είναι ή περιέχει κλήση συστήματος (system call)
   2. αν η διεργασία δε τελειώσει εγκαίρως στο κβάντο χρόνου της
2. αν συμβεί page fault και ο κώδικας οδηγεί σε εξαίρεση null pointer dereference
3. δεν έχω άδεια πρόσβασης/exception (π.χ. διαίρεση με το 0, null pointer)
4. συμβαίνει ένα cache miss
   1. αν η σελίδα βρίσκεται στο page table τότε το ΛΣ δεν εμπλέκεται
   2. αν όχι τότε εμπλέκεται στη μεταφορά της από το δίσκο
5. συμβαίνει ένα TLB (Translation Lookaside buffer) miss
   1. τότε επιχειρείται ανάγνωση του page table
   2. το ΛΣ εμπλέκεται αν γίνει page fault
6. συμβαίνει ένα σφάλμα σελίδας (page fault)
   1. ελέγχεται το TLB -> μετά το page table -> τότε page fault
   2. το ΛΣ ξυπνάει αφού έγινε αναφορά σε άκυρη σελίδα (invalid)
   3. το ΛΣ συμβουλεύεται το χάρτη μνήμης και έλεγχος δικαιωμάτων
      1. μη επιτρεπόμενη πρόσβαση -> segmentation fault
      2. επιτρεπόμενη πρόσβαση αλλά invalid
         1. η σελίδα είναι στο δίσκο
         2. ο χρονοδρομολογητής βάζει την διεργασία σε WAITING και τη βγάζει από τη CPU συνεχίζοντας με την επόμενη που βρίσκεται στην ουρά αναμονής
         3. μεταφορά από το δίσκο
         4. Αναζήτηση στη μνήμη για ελεύθερο πλαίσιο. Αν υπάρχει φόρτωση της καινούριας σελίδας. Αν δεν υπάρχει εύρεση πλαισίου για αντικατάσταση (FIFO/LRU)
         5. ανάγνωση δεδομένων
         6. ανανέωση χάρτη μνήμης
         7. ανανέωση page table (invalid -> valid)
         8. χρονοδρομολογητής -> READY state
         9. διακόπτεται η CPU και εκτελείται
7. το πρόγραμμα του χρήστη κάνει διαίρεση με το 0
   1. το ΛΣ στέλνει segmentation fault και σκοτώνει τη διεργασία εκτός αν έχουμε signal handler
8. το πρόγραμμα του χρήστη κάνει null pointer dereference
   1. το ΛΣ στέλνει segmentation fault και σκοτώνει τη διεργασία εκτός αν έχουμε signal handler
9. το πρόγραμμα του χρήστη κάνει άλμα σε χώρο μνήμης που περιέχει δεδομένα
   1. θα γίνει page fault και το ΛΣ θα τερματίσει το πρόγραμμα αφού σε αυτό το χώρο δε θα υπάρχει δικαίωμα “x”
10. το πρόγραμμα του χρήστη προσπαθεί να μπει σε κρίσιμο τμήμα
    1. το ΛΣ εμπλέκεται στη περίπτωση που χρησιμοποιούνται semaphores /mutexes και το κρίσιμο τμήμα δεν είναι διαθέσιμο με τη wait/wakeup οπότε η διεργασία κάνει sleep και χρονοδρομολογείται άλλη διεργασία

Τι χαρακτηριστικά πρέπει να έχει ένας κώδικας (ή αλγόριθμος) ώστε η ταυτόχρονη εκτέλεσή του από πολλές διεργασίες (ή νήματα) να μπορεί να οδηγήσει σε ασυνέπειες; Αναφέρετε ένα παράδειγμα. -> Αν υπάρχουν πολλαπλές διεργασίες που εκτελούν εργασίες σε κοινά δεδομένα και ο χρήστης δεν έχει προνοήσει για κατάλληλο συγχρονισμό αυτών, δηλαδή οι διεργασίες να εισέρχονται μέσω αμοιβαίου αποκλεισμού στο κρίσιμο τμήμα και όχι ταυτόχρονα. τότε μπορεί να υπάρξουν ασυνέπειες. Χαρακτηριστικό παράδειγμα η αυξομείωση μιας ακέραιας μεταβλητής που ακολουθεί το μοντέλου παραγωγού-καταναλωτή. Αν επιτρέπεται στις διεργασίες να εισάγονται ταυτόχρονα στο κρίσιμο τμήμα ( αύξηση/μείωση της μεταβλητής ) τότε η μία δεν γνωρίζει την μεταβολή που θα επιφέρει η άλλη με συνέπεια στο τελικό επιθυμητό αποτέλεσμα.

Κάθε επεξεργαστής διατηρεί ξεχωριστή ουρά έτοιμων διεργασιών: 1. Απλή υλοποίηση 2. Ελάχιστος χρόνος αναμονής 3. αποφεύγεται η μετακίνηση διεργασιών μεταξύ τον επεξεργαστών 4. Οδηγούμαστε, όμως, σε προσκόλληση διεργασιών σε επεξεργαστή (processor affinity) 5. κάποιος επεξεργαστής μπορεί να μείνει ανενεργός για μεγάλο διάστημα, εφ‘ όσον δεν του ανατίθενται καινούριες διεργασίες.

Η καθολική ουρά: 1. μπορεί να το αντιμετωπίσει 2. χρειάζεται όμως προσοχή στον δρομολογητή ώστε να αποφεύγεται η ανάθεση της ίδιας διεργασίας σε πολλούς επεξεργαστές.3. Επίσης δημιουργούνται κοινά δεδομένα στις caches των επεξεργαστών, αφού λειτουργούν την ίδια διεργασία (σε διακοπτό χρονοδρομολογητή).

**Διεργασίες: 1.**είναι ένα ζωντανό πρόγραμμα που εκτελείται ΤΩΡΑ **2.** για να γίνει ζωντανό πρέπει να έχει μνήμη **ΚΑΙ** αρχιτεκτονική κατάσταση (στιγμιότυπο μίας διεργασίας στη CPU, οι τιμές των καταχωρητών της) **3.** Κάθε διεργασία έχει τη δικιά της μνήμη -> τον δικό της χώρο διευθύνσεων μνήμης, πίνακα προγραμμάτων και πόρους, όπως αρχεία και συσκευές εισόδου/εξόδου. **4.** Μπορεί να περιέχει πολλά νήματα **5.** Είναι απομονωμένες μεταξύ τους **(**δεν μπορούν να αλληλεπιδρούν άμεσα μεταξύ) **5.** Ασφάλεια: Δε μοιράζονται μνήμη οπότε η παραβίαση των δεδομένων είναι δυσκολότερη **Νήματα: 1.** Είναι ένα εκτελέσιμο μέρος μιας διεργασίας **2.** Έχει κοινή μνήμη με τα υπόλοιπα νήματα της ίδιας διεργασίας **-** μπορούν να επικοινωνούν μεταξύ τους ευκολότερα **-** μοιράζεται τον ίδιο χώρο διευθύνσεων μνήμης, πίνακα προγραμμάτων και πόρους  **3.** Απόδοση: Τα νήματα είναι πιο ελαφριά και γρήγορα στη δημιουργία και την εναλλαγή από τις διεργασίες.

Εκτελέσιμο που μεταγλωττίζεται για το Linux δεν θα εκτελεστεί σωστά σε άλλα λειτουργικά συστήματα, όπως τα Windows ή το macOS, λόγω των διαφορετικών τεχνολογιών και διεπαφών που χρησιμοποιούνται από αυτά τα λειτουργικά συστήματα ->προβλήματα στις κλήσεις του συστήματος (system calls)

π.χ. **ld r1, [r2]** μεταφερει το περιεχομενο της θεσης μνημης στην οποία δειχνει ο καταχωητης r2 στον καταχωρητη r1 και οδηγει:

1. στην εμπλοκη του ΛΣ δυο φορες -> Page fault -> μεταφορά από το δίσκο -> εκτέλεση

2. τερματισμός διεργασίας -> χάρτης μνήμης -> έλεγχος δικαιωμάτων -> δεν έχουμε άδεια για read στο r2 -> οπότε το ΛΣ τερματίζει τη διεργασία

3. το πρόγραμμα συνέχισε -> χάρτης μνήμης -> έλεγχος δικαιωμάτων -> δεν έχουμε άδεια για read στο r2 -> invalid -> οπότε μεταφορά από το δίσκο -> valid στο χάρτη μνήμης -> εκτέλεση εντολής

Μηχανισμούς υλικού (hardware) που υποστηρίζουν:

1. το διαχωρισμό της εκτέλεσης του ΛΣ από τα προγράμματα χρήστη
   1. με προνόμια -> κάποιες διεργασίες πρέπει να εκτελούνται μόνο από ΛΣ
   2. γίνεται διαχωρισμός του επεξεργαστή σε δύο καταστάσεις kernel-user
   3. αυτό γίνεται με τη δέσμευση ενός bit σε έναν καταχωρητή -> bit mode
2. τη χρονοδρομολόγηση
   1. ο καταχώρησης που λειτουργεί ως counter και μετρά το κβάντο χρόνου, χρόνο που βρίσκεται μία διεργασία τη CPU
3. την εικονική μνήμη
   1. TLB (Translation Lookaside buffer) είναι μία cache μνήμης που βοηθάει στη μετάφραση διευθύνσεων
   2. εκεί αποθηκεύονται οι διευθύνσεις των πιο πρόσφατων προσπελάσεων στο page table (μαζί με τις γειτονικές τους διευθύνσεις – γιατί έχουν μεγάλη πιθανότητα να προσπελαστούν στη συνέχεια)
   3. page table register -> που αποθηκεύει τη πρώτη διεύθυνση ου page table για να μπορείς να βρεις πού είναι
4. το συγχρονισμό
   1. οι ατομικές εντολές

**Ανάγκη ύπαρξης** ΛΣ:

* διαχείριση διεργασιών (pipes, αρχεία) – νημάτων
* συγχρονισμό, απομόνωση και ασφάλεια διεργασιών
* διαχείριση μνήμης (φυσικής - εικονικής)
* διαχείριση αρχείων
* επιτρέπει την επικοινωνία με συσκευές εισόδου/εξόδου (ποντίκι, πληκτρολόγιο)
* επιτρέπει τη δικτύωση του υπολογιστή (IP address, WIFI)
* επικοινωνία με τον χρήστη (command line/ user interface)