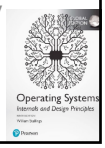
ΕΠΛ222: Λειτουργικά Συστήματα

(μετάφραση στα ελληνικά των διαφανειών του βιβλίου Operating Systems: Internals and Design Principles, 9/E, William Stallings)

Ενότητα 7 (Κεφάλαιο 7) Διαχείριση Μνήμης

Οι διαφάνειες αυτές έχουν συμπληρωματικό και επεξηγηματικό χαρακτήρα και σε καμία περίπτωση δεν υποκαθιστούν το βιβλίο

Γιώργος Α. Παπαδόπουλος
Τμήμα Πληροφορικής
Πανεπιστήμιο Κύπρου



1

Περιεχόμενα

- Βασικές αρχές διαχείρισης μνήμης.
- Διαμοίραση μνήμης.
- Σελιδοποίηση και κατάτμηση.

2

Ανάγκη για διαχείριση μνήμης

- Η μνήμη είναι φτηνή στις μέρες μας και γίνεται φτηνότερη.
  - Όμως αυξάνεται και η ανάγκη των εφαρμογών σε περισσότερη μνήμη.
  - Όση μνήμη και να έχουμε δεν είναι αρκετή!
- Σε ένα σύστημα μονοπρογραμματισμού, η μνήμη χωρίζεται σε αυτή που χρησιμοποιεί το Λ.Σ. και σε αυτή που χρησιμοποιεί το πρόγραμμα του χρήστη.
- Αν υποστηρίζεται πολυπρογραμματισμός, τότε η μνήμη για τον χρήστη διαμοιράζεται ανάμεσα στις εκτελούμενες διεργασίες.
- Λόγω της μεγάλης διαφοράς ταχύτητας μεταξύ της ΚΜΕ και των συσκευών Ε/Ε, είναι σημαντικό να υπάρχει πάντα στη μνήμη ένας ικανός αριθμός διεργασιών έτσι ώστε να μην υποχρησιμοποιείται η ΚΜΕ αν όλες οι διεργασίες που βρίσκονται στη μνήμη είναι σε αναστολή.
- Κατ' επέκταση το Λ.Σ. πρέπει να εναλλάσσει με έξυπνο τρόπο τις διεργασίες που βρίσκονται στη μνήμη με άλλες που βρίσκονται στο δίσκο, έτσι ώστε να υπάρχει πάντα στη μνήμη ένας ικανοποιητικός αριθμός από διεργασίες που είναι σε κατάσταση έτοιμες για εκτέλεση.

3

Διαχείριση μνήμης

- Το τμήμα του Λ.Σ. που διαχειρίζεται τη μνήμη λέγεται Διαχειριστής Μνήμης (Memory Manager).
- Καθήκον του είναι να παρακολουθεί ποια τμήματα της μνήμης είναι σε χρήση και ποια όχι, να χορηγεί μνήμη σε διεργασίες όποτε τη χρειάζονται και να την επαναχορηγεί σε άλλες όταν οι πρώτες ολοκληρώνουν την αποστολή τους, καθώς και να διαχειρίζεται την εναλλαγή πληροφοριών (swapping) μεταξύ κύριας μνήμης και δίσκου όταν η κύρια μνήμη δεν είναι αρκετή για να εξυπηρετηθούν όλες οι διεργασίες.

4

Ανάγκες για τη διαχείριση μνήμης

- Συγκεκριμένα, η διαχείριση μνήμης απαιτεί την ικανοποίηση των ακόλουθων αναγκών:
  - Μετατόπιση (relocation).
  - Προστασία (protection).
  - Διαμοίραση (sharing).
  - Λογική οργάνωση (logical organisation).
  - Φυσική οργάνωση (physical organisation).

5

Μετατόπιση

- Αν μία διεργασία χρειασθεί να απομακρυνθεί από την κύρια μνήμη, δεν θα πρέπει να είναι αναγκαίο όταν επανέλθει να αποθηκευθεί στον ίδιο χώρο που καταλάμβανε προηγουμένως.
- Για να ικανοποιηθεί αυτή η ανάγκη θα πρέπει οι διευθύνσεις μνήμης που χρησιμοποιούν οι εντολές του προγράμματος να μην είναι απόλυτες αλλά σχετικές ως προς το περιεχόμενο κάποιου καταχωρητή.
- Κατά την αποκωδικοποίησή τους, οι διευθύνσεις μνήμης μετατρέπονται από σχετικές σε απόλυτες (φυσικές).

6

Σχετικές διευθύνσεις μνήμης

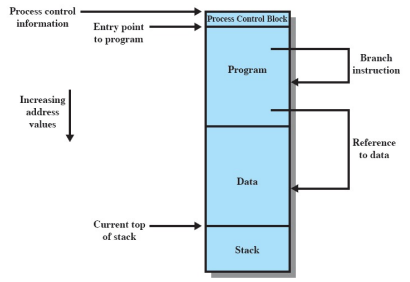


Figure 7.1 Addressing Requirements for a Process

7

Προστασία

- Καμία διεργασία δεν πρέπει να μπορεί να αναφερθεί σε θέση μνήμης που ανήκει σε άλλη διεργασία (χωρίς σχετική άδεια).
- Το πρόβλημα γίνεται πιο δύσκολο λόγω της μετατόπισης των διεργασιών στην κύρια μνήμη και ο έλεγχος πρέπει να γίνεται δυναμικά και όχι στατικά (την ώρα της μετάφρασης).

8

Διαμοίραση

- Περισσότερες από μία διεργασίες θα πρέπει να μπορούν να αναφέρονται στον ίδιο χώρο μνήμης.
  - Π.χ. όλες οι διεργασίες ενός προγράμματος θα πρέπει να μπορούν να αναφέρονται στο χώρο μνήμης στον οποίο βρίσκεται αποθηκευμένος ο κώδικας του προγράμματος.
- Είναι επιθυμητό όλες οι διεργασίες να έχουν πρόσβαση στο ίδιο αντίγραφο του προγράμματος αντί η κάθε μία να έχει πρόσβαση σε διαφορετικό αντίγραφο.

9

Λογική Οργάνωση

- Αν και η μνήμη (κύρια και περιφερειακή) είναι ουσιαστικά ένας μονοδιάστατος πίνακας που αποτελείται από μία σειρά από bytes, εν τούτοις σε λογικό επίπεδο θα πρέπει να υποστηρίζει τη λογική δομή ενός τυπικού προγράμματος:
  - Χωρισμός σε ενότητες (modules), κοινή χρήση κάποιων ενοτήτων από προγράμματα, κλπ.
- Αυτή η προσέγγιση έχει ένα αριθμό από πλεονεκτήματα, όπως:
  - Μερική αλλαγή και μετάφραση κάποιων ενοτήτων.
  - Χρήση διαφορετικών μηχανισμών προστασίας σε κάθε ενότητα (π.χ. μόνο για διάβασμα ή εκτέλεση).

10

Φυσική Οργάνωση

- Δεν είναι λογικό να επιφορτίσουμε τον προγραμματιστή με την ευθύνη να διαχειρίζεται τη μνήμη.
- Η διαθέσιμη μνήμη για κάποιο πρόγραμμα μπορεί να μην είναι αρκετή για την εκτέλεση του προγράμματος και οι τεχνικές επικάλυψης που πρέπει να χρησιμοποιηθούν σε τέτοιες περιπτώσεις είναι χρονοβόρες.
- Ο προγραμματιστής γενικώς δεν είναι σε θέση να γνωρίζει πόση μνήμη είναι διαθέσιμη.
- Επομένως, η ευθύνη μεταφοράς πληροφοριών μεταξύ κύριας και περιφερειακής μνήμης πρέπει να ανήκει στο Λ.Σ.

11

Περιεχόμενα

- Βασικές αρχές διαχείρισης μνήμης.
- Διαμοίραση μνήμης.
- Σελιδοποίηση και κατάτμηση.

12

Διαμοίραση

- Μία από τις πρώτες τεχνικές διαχείρισης της μνήμης.
- Αν και δεν χρησιμοποιείται τόσο πολύ στις μέρες μας (χωρίς συνδυασμό με ιδεατή μνήμη), η μελέτη της θα βοηθήσει στην κατανόηση πιο μοντέρνων τεχνικών, όπως αυτών που βασίζονται σε ιδεατή μνήμη.

13

13

Είδη διαμοίρασης

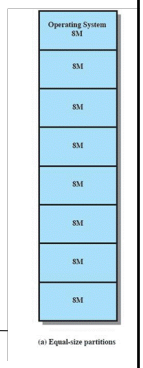
- Σταθερά τμήματα (fixed partitioning).
- Δυναμικά τμήματα (dynamic partitioning).
- Απλή σελιδοποίηση (simple paging).
- Απλή κατάτμηση (simple segmentation).
- Σελιδοποίηση με χρήση ιδεατής μνήμης (virtual memory paging).
- Κατάτμηση με χρήση ιδεατής μνήμης (virtual memory segmentation).

14

14

Σταθερά Τμήματα

- Με εξαίρεση το χώρο της μνήμης που χρειάζεται το Λ.Σ., ο υπόλοιπος χωρίζεται σε σταθερά ισομεγέθη τμήματα.
- Οποιαδήποτε διεργασία της οποίας το μέγεθος είναι μικρότερο ή ίσο με αυτό των τμημάτων, μπορεί να φορτωθεί σε οποιοδήποτε ελεύθερο τμήμα.
- Το Λ.Σ. μπορεί να απομακρύνει μία διεργασία από κάποιο τμήμα αν αυτή δεν εκτελείται ή δεν είναι σε κατάσταση έτοιμη για εκτέλεση.



(a) Equal-size partitions

15

15

Προβλήματα με την τεχνική των ισομεγεθών σταθερών τμημάτων — 1

- Στην περίπτωση που κάποιο πρόγραμμα είναι μεγαλύτερο από το μέγεθος των τμημάτων θα πρέπει να χωρισθεί σε τμήματα επικάλυψης (overlays) και να μεταφέρεται στο χώρο της κύριας μνήμης που ανήκει στο τμήμα εκείνο του προγράμματος που πρέπει να εκτελεσθεί, επικαλύπτοντας το προηγούμενο τμήμα του προγράμματος.
- Εδώ γίνεται προσπάθεια να χωρισθεί ένα πρόγραμμα σε μέρη ανεξάρτητα (κατά το δυνατόν) μεταξύ τους: π.χ. ένα τμήμα για είσοδο δεδομένων, ένα για την επεξεργασία τους και ακόμα ένα για την έξοδο των αποτελεσμάτων.
- Η εργασία αυτή πρέπει να γίνει από τον προγραμματιστή και είναι βασανιστική, βαρετή και χρονοβόρα.

16

16

Προβλήματα με την τεχνική των ισομεγεθών σταθερών τμημάτων — 2

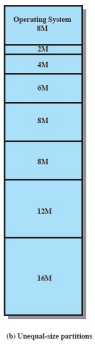
- Επίσης, η μέθοδος αυτή οδηγεί σε σπατάλη μνήμης στην (όχι σπάνια) περίπτωση που κάποιο πρόγραμμα είναι μικρότερο από το μέγεθος των τμημάτων.
- Αυτό το φαινόμενο ονομάζεται εσωτερικός κατακερματισμός (internal fragmentation).

17

17

Λύση: Ανισομεγέθη Σταθερά Τμήματα

- Ελαττώνει το πρόβλημα αλλά δεν το εξαλείφει τελείως.
- Στο διπλανό σχήμα, προγράμματα μέχρι 16 MB μπορούν να αποθηκευθούν χωρίς χρήση επικάλυψης και τα μικρότερα προγράμματα μπορούν να αποθηκευθούν σε μικρότερα τμήματα, μειώνοντας έτσι τον εσωτερικό κατακερματισμό.



(b) Unequal-size partitions

18

18

Αλγόριθμοι Τοποθέτησης για τα σταθερά ισομεγέθη τμήματα

- Ο αλγόριθμος τοποθέτησης (placement algorithm) για τα σταθερά ισομεγέθη τμήματα είναι πολύ απλός:
  - Το πρόγραμμα τοποθετείται σε οποιοδήποτε ελεύθερο τμήμα.

19

19

Αλγόριθμοι Τοποθέτησης για τα σταθερά ανισομεγέθη τμήματα — 1

- Σε αυτήν την περίπτωση τίθεται το ερώτημα με ποιο τρόπο διαμοιράζουμε τα διαθέσιμα τμήματα στα προγράμματα. Υπάρχουν δύο εναλλακτικές λύσεις:
- Κάνοντας χρήση πολλαπλών ουρών αναθέτουμε στο κάθε πρόγραμμα το μικρότερο σε μέγεθος τμήμα που ικανοποιεί τις ανάγκες του.
  - Αυτό προϋποθέτει όμως ότι κάποιο πρόγραμμα γνωρίζει εκ των προτέρων τις ανάγκες του σε μνήμη.
  - Επίσης, μπορεί να ελαχιστοποιείται η σπατάλη μνήμης μέσα σε ένα τμήμα αλλά συνολικά υπάρχει ακόμα σπατάλη γιατί μπορεί κάποιο τμήμα να παραμένει αχρησιμοποίητο αν δεν υπάρχει κάποιο πρόγραμμα αρκετά μεγάλο για να δικαιολογείται η ανάθεσή του σε αυτό.

20

20

Αλγόριθμοι Τοποθέτησης για τα σταθερά ανισομεγέθη τμήματα — 2

- Εναλλακτικά, κάνοντας χρήση μίας ουράς, αναθέτουμε σε κάποιο πρόγραμμα το μικρότερο διαθέσιμο τμήμα.
  - Αν δεν υπάρχει διαθέσιμο τμήμα, κάποιο από τα προγράμματα που βρίσκονται στην κύρια μνήμη θα πρέπει να μεταφερθεί στο δίσκο (κατά προτίμηση κάποιο που βρίσκεται σε τμήμα του μεγέθους που χρειάζεται το νέο πρόγραμμα για να τρέξει, αλλά μπορεί να ληφθούν υπ' όψη εδώ και άλλα κριτήρια όπως προτεραιότητες, κατάσταση διεργασιών, κλπ.).

21

21

Τοποθέτηση σε σταθερά τμήματα

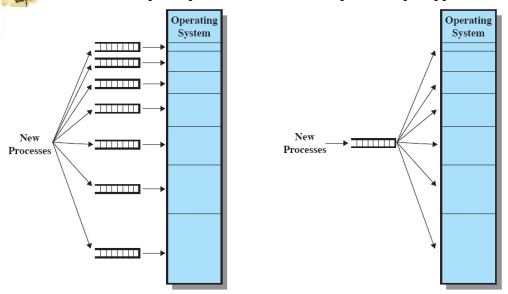


Figure 7.3 Memory Assignment for Fixed Partitioning

22

22

Παραμένοντα προβλήματα με την τεχνική των σταθερών τμημάτων

- Ο αριθμός των εν ενεργεία διεργασιών στο σύστημα περιορίζεται στον αριθμό των τμημάτων.
- Επειδή γενικώς οι ανάγκες σε μνήμη μίας τυπικής διεργασίας δεν μπορούν να προκαθορισθούν, ο χωρισμός της μνήμης σε τμήματα διαφορετικού μεγέθους δεν μπορεί να γίνει αποτελεσματικά και σπαταλείται σημαντική ποσότητα μνήμης.
- Η ύπαρξη πολλών διεργασιών περιορισμένης ανάγκης σε μνήμη, επίσης θα οδηγήσει σε σπατάλη μνήμης.
- Ένα από τα πιο επιτυχημένα Λ.Σ. που χρησιμοποιούσαν αυτή τη μέθοδο είναι το OS/MFT (Multiprogramming with a Fixed number of Tasks) της IBM.

23

23

Δυναμικά Τμήματα

- Μερικά από τα προβλήματα της χρήσης σταθερών τμημάτων επιλύονται με τη χρήση μεταβλητών τμημάτων.
- Σε κάθε διεργασία εκχωρείται η ποσότητα της μνήμης που χρειάζεται, εφ' όσον βεβαίως είναι διαθέσιμη.
- Κάποια στιγμή, κάποια από τις διεργασίες που βρίσκονται στην κύρια μνήμη θα αποδεσμεύσει τη μνήμη που χρησιμοποιούσε (είτε γιατί τερμάτισε είτε γιατί θα μεταφερθεί στο δίσκο).
- Τον χώρο που απελευθερώνεται μπορεί να καταλάβει μία *ή περισσότερες* διεργασίες ανάλογα με τις ανάγκες τους.
- Αντίθετα με την περίπτωση των σταθερών τμημάτων, εδώ ο αριθμός, η θέση και το μέγεθος των τμημάτων καθώς επίσης και ο αριθμός των διεργασιών στην κύρια μνήμη είναι μεταβλητός.

24

24

Δυναμικά Τμήματα: Πλεονεκτήματα

- Το βασικό πλεονέκτημα της μεθόδου αυτής είναι ότι δεν εγκλωβιζόμαστε σε σταθερό αριθμό τμημάτων και αυξάνεται έτσι η απόδοση της μνήμης μέσω της αύξησης του βαθμού πολυπρογραμματισμού του συστήματος.

25

25

Δυναμικά Τμήματα: Μειονεκτήματα — 1

- Από την άλλη πλευρά όμως ο μηχανισμός δέσμευσης και αποδέσμευσης της μνήμης και ο μηχανισμός παρακολούθησης των μεταβολών που λαμβάνουν χώρα γίνεται πιο πολύπλοκος.

26

26

Δυναμικά Τμήματα: Μειονεκτήματα — 2

- Ένα άλλο πρόβλημα που δημιουργείται είναι ότι κάποια στιγμή θα υπάρχουν στη μνήμη πολλά μικρά κενά τμήματα τα οποία αν και συνδυασμένα θα μπορούσαν να ικανοποιήσουν τις ανάγκες κάποιας διεργασίας, από μόνα τους είναι άχρηστα.
- Αυτό το φαινόμενο ονομάζεται εξωτερικός κατακερματισμός (external fragmentation).
- Ο Knuth υπολόγισε ότι σε γενικές γραμμές τα κενά h θα είναι κατά μέσο όρο τα μισά από ότι οι διεργασίες n (δηλαδή h=n/2). Το αποτέλεσμα αυτό είναι γνωστό σαν *ο κανόνας του 50%* (fifty percent rule) και βασίζεται στην παρατήρηση ότι εφαπτόμενα κενά συνενώνονται σε ένα.

27

27

Δυναμικά Τμήματα: Μειονεκτήματα — 3

- Μία πιθανή λύση θα ήταν η περιοδική μετακίνηση από το Λ.Σ. όλων των τμημάτων έτσι ώστε να βρίσκονται σε συνεχόμενες θέσεις και επομένως οι ελεύθεροι χώροι να είναι και αυτοί μαζί σαν ένα τμήμα.
- Αυτή η μέθοδος ονομάζεται συμπίεση μνήμης (compaction) αλλά χρησιμοποιείται σπάνια γιατί είναι πολύ δαπανηρή σε χρόνο ΚΜΕ.
  - Π.χ. σε μία μηχανή με 1 MB μνήμη και με δυνατότητα μεταφοράς 1 byte/msec (δηλαδή 1 MB / sec) θα χρειαζόταν ένα δευτερόλεπτο για τη συμπίεση όλης της μνήμης.

28

28

Παράδειγμα με δυναμικά τμήματα

- *Δείχνει εικονογραφικά και το πρόβλημα του εξωτερικού κατακερματισμού.*

Αναφορά στο σχήμα 7.4

29

29

Αλγόριθμοι Τοποθέτησης για τα δυναμικά τμήματα

- Για την επιλογή του κατάλληλου τμήματος μνήμης για τις ανάγκες μίας διεργασίας χρησιμοποιείται ένας από τους ακόλουθους αλγόριθμους τοποθέτησης.
- Ο κοινός παρανομαστής για όλους είναι η προσπάθεια εύρεσης ενός τμήματος μνήμης ίσου ή μεγαλύτερου αυτού που κάποια διεργασία έχει ανάγκη με τρόπο που να ελαχιστοποιεί τη σπατάλη μνήμης αλλά και τη συχνότητα χρήσης τεχνικών συμπίεσης.

30

30

Πρώτη τοποθέτηση

- Ο αλγόριθμος της πρώτης τοποθέτησης (first-fit) σαρώνει τη μνήμη και διαλέγει το πρώτο τμήμα που θα βρει και είναι κατάλληλο για τις ανάγκες του προγράμματος.
- Αν το τμήμα αυτό είναι μεγαλύτερο από όσο χρειάζεται το διασπά σε δύο κομμάτια.
- Είναι ο απλούστερος από τους αλγόριθμους, γρήγορος στην εκτέλεση και παραδόξως έχει συνήθως καλά αποτελέσματα.

Επόμενη τοποθέτηση

- Μία παραλλαγή του προηγούμενου αλγόριθμου είναι αυτός της επόμενης τοποθέτησης (next-fit) όπου η αναζήτηση ενός κατάλληλου τμήματος ξεκινάει από το σημείο που βρέθηκε το τελευταίο κατάλληλο τμήμα αντί από την αρχή της μνήμης.
- Μελέτες προσομοίωσης από τον Bays έδειξαν ότι ο αλγόριθμος αυτός έχει ελαφρώς χειρότερα αποτελέσματα από αυτόν της πρώτης τοποθέτησης.
- Τείνει να χορηγεί μνήμη από το μεγαλύτερο διαθέσιμο κομμάτι που βρίσκεται προς το τέλος της μνήμης (υψηλότερες διευθύνσεις).
- Επομένως η περιοχή εκείνη της μνήμης σπάει γρήγορα σε μικρά και άχρηστα τμήματα, δημιουργώντας την ανάγκη χρήσης τεχνικών συμπίεσης.
- Παρεμπιπτόντως, το αντίθετο συμβαίνει με τον προηγούμενο αλγόριθμο ο οποίος δημιουργεί το ίδιο πρόβλημα στις χαμηλές θέσεις μνήμης.

Καλύτερη τοποθέτηση

- Ο αλγόριθμος της καλύτερης τοποθέτησης (best-fit) προσπαθεί να βρει το τμήμα εκείνο της μνήμης που είναι όσο γίνεται πιο μικρό (αλλά βεβαίως να εξακολουθεί να ικανοποιεί τις ανάγκες του προγράμματος).
- Αν και φαίνεται λογική αυτή η προσέγγιση, γιατί προσπαθεί να ελαχιστοποιήσει τα άχρηστα κενά, εν τούτοις δημιουργεί μεγαλύτερο πρόβλημα γιατί τα κενά που αναπόφευκτα θα δημιουργηθούν είναι πολύ μικρά και άχρηστα ακόμα και για μικρά προγράμματα.
- Επίσης είναι πιο αργός από τους προηγούμενους λόγω της σάρωσης όλης της μνήμης.

Χειρότερη τοποθέτηση

- Ο αλγόριθμος της χειρότερης τοποθέτησης (worst-fit) κάνει ακριβώς το αντίθετο ψάχνοντας να βρει το μεγαλύτερο τμήμα έτσι ώστε όταν διασπασθεί το υπόλοιπο κομμάτι να είναι αρκετά μεγάλο για να παραμείνει χρήσιμο.
- Προγράμματα προσομοίωσης έδειξαν ότι δεν είναι πολύ αποτελεσματικός.

Γρήγορη τοποθέτηση

- Τέλος, ο αλγόριθμος της γρήγορης τοποθέτησης (quick-fit) αποθηκεύει τα πιο συνηθισμένα μεγέθη μνήμης που ζητούνται.
- Αν και η ζήτηση ενός τμήματος προκαθορισμένου μεγέθους είναι πολύ γρήγορη, εν τούτοις ο αλγόριθμος υποφέρει από τα ίδια προβλήματα με τους υπόλοιπους.

Εφαρμογή μερικών αλγορίθμων

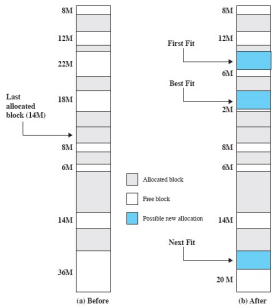


Figure 7.5 Example Memory Configuration before and after Allocation of 16-Mbyte Block

Αντικατάσταση διεργασιών

- Ακόμα και με τη χρήση της τεχνικής της συμπίεσης, θα φτάσει κάποια στιγμή όπου όλη η μνήμη θα είναι δεσμευμένη από διεργασίες που βρίσκονται σε κατάσταση αναστολής.
- Σε αυτήν την περίπτωση το Λ.Σ. είναι υποχρεωμένο να μεταφέρει κάποιες από αυτές τις διεργασίες στην περιφερειακή μνήμη και να τις αντικαταστήσει με άλλες από εκεί, που είναι έτοιμες για εκτέλεση.
- Οι πολιτικές με βάση τις οποίες το Λ.Σ. αποφασίζει πως θα γίνονται τέτοιου είδους αντικαταστάσεις αποτελούν τους αλγόριθμους αντικατάστασης και θα εξετασθούν αργότερα σε συνάρτηση με την τεχνική της ιδεατής μνήμης.

Το Σύστημα των Φίλων — 1

- Το Σύστημα των Φίλων (buddy system) επινοήθηκε από τους Knuth και Knowlton, σαν μία μέση λύση μεταξύ των τεχνικών των σταθερών και μεταβλητών τμημάτων.
- Η μνήμη μπορεί να διαιρεθεί σε μπλοκ μεγέθους 2^N bytes, δηλαδή μεγέθους 1, 2, 4, 8, 16 … bytes μέχρι το συνολικό μέγεθός της.
- Ας θεωρήσουμε ότι η διαθέσιμη μνήμη είναι 1 MB.
- Αν τώρα η πρώτη αίτηση για μνήμη είναι των 100 K, τότε το πιο κοντινό στις ανάγκες της αίτησης αυτής μέγεθος μπλοκ είναι 128 K.

Το Σύστημα των Φίλων — 2

- Επομένως, το αρχικό μπλοκ του 1 MB σπάει σε δύο "φίλους" των 512 K, ένα από αυτά σπάει περαιτέρω σε δύο άλλους "φίλους" των 128 K και ένα από αυτά τα τελευταία χρησιμοποιείται για την αίτηση.
- Αν η δεύτερη αίτηση έχει ανάγκη σε 256 K, παίρνει αμέσως το υπάρχων μπλοκ αυτού του μεγέθους.
- Η ευελιξία της μεθόδου αυτής βρίσκεται στο γεγονός ότι αν ολοκληρώσουν την εκτέλεσή τους δύο διεργασίες που καταλαμβάνουν γειτονικούς χώρους, τα δύο αντίστοιχα φιλικά μπλοκ συνενώνονται σε ένα, επιτρέποντας έτσι την εξυπηρέτηση επιπλέον αιτήσεων.
- Η τεχνική αυτή αντιμετωπίζει με σχετική επιτυχία τα μειονεκτήματα των τεχνικών των σταθερών και μεταβλητών τμημάτων.
- Αν και η τεχνικές που βασίζονται σε ιδεατή μνήμη είναι καλύτερες, το σύστημα των φίλων εφαρμόζεται με επιτυχία σε παράλληλα συστήματα και στον πυρήνα του Unix.

Παράδειγμα

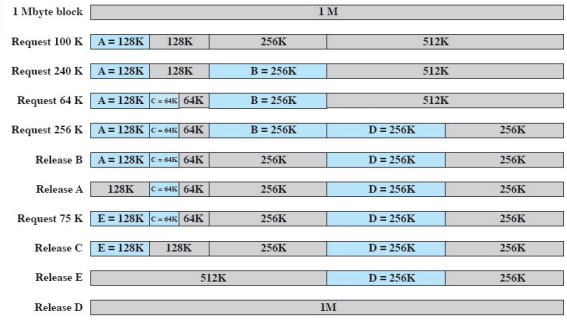


Figure 7.6 Example of Buddy System

Αναπαράσταση σε δενδροειδή μορφή μετά την απελευθέρωση μνήμης από τη διεργασία B

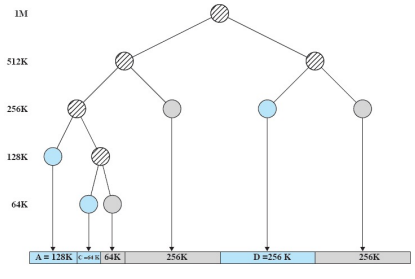


Figure 7.7 Tree Representation of Buddy System

Μετατόπιση

- Για τους περισσότερους από τους μηχανισμούς που εξετάσαμε μέχρι τώρα ισχύει ότι το τμήμα μνήμης που θα καταλάβει μία διεργασία στην κύρια μνήμη δεν είναι κατ' ανάγκη το ίδιο με αυτό που καταλάμβανε την προηγούμενη φορά που είχε εισέλθει στην κύρια μνήμη (συγκρίνεται με το μηχανισμό του CTSS).
- Επίσης, κατά τη συμπίεση της μνήμης, μία διεργασία τυχόν να μετακινηθεί σε άλλο χώρο.
- Θα πρέπει λοιπόν οι διευθύνσεις μνήμης που χρησιμοποιούν οι εντολές του προγράμματος να αλλάζουν ανάλογα.
- Αυτό οδηγεί στην ανάγκη υποστήριξης μηχανισμών μετατόπισης (relocation).

Είδη διευθύνσεων

- Λογική.
  - Αναφορά σε μία θέση μνήμης ανεξάρτητα από το που στη μνήμη βρίσκονται τα δεδομένα.
- Σχετική.
  - Αναφορά σε μία θέση μνήμης με βάση ένα γνωστό σημείο αναφοράς.
- Φυσική ή απόλυτη.
  - Αναφορά στην απόλυτη θέση μνήμης στο φυσικό υλικό χώρο μνήμης του συστήματος.

43

43

Μηχανισμός μετατόπισης

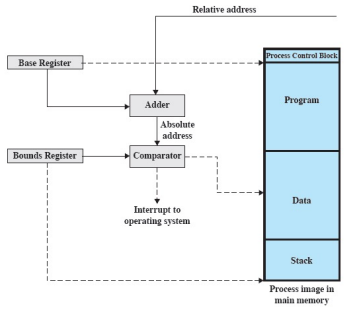


Figure 7.8 Hardware Support for Relocation

44

44

Χρήση καταχωρητών στη μετατόπιση

- Η τεχνική που συνήθως ακολουθείται (και η οποία ικανοποιεί και τις ανάγκες της προστασίας) είναι να υπάρχουν δύο ειδικοί καταχωρητές:
  - ο καταχωρητής βάσης (base register), και
  - ο καταχωρητής ορίου (limit, bounds register).
- Ο πρώτος έχει ως τιμή τη διεύθυνση της αρχής του τμήματος ενώ ο δεύτερος το μέγεθος (ή το τέλος) του τμήματος.
- Σε κάθε διεύθυνση των εντολών της διεργασίας προστίθεται αυτόματα η τιμή του καταχωρητή βάσης και η καινούργια διεύθυνση ελέγχεται με βάση την τιμή του καταχωρητή ορίου για να διαπιστωθεί αν ανήκει στα επιτρεπτά όρια μνήμης που ελέγχει η διεργασία.
- Έτσι αν ο καταχωρητής βάσης έχει τιμή 130K και το πρόγραμμα έχει την εντολή CALL 100, τότε αυτή μετατρέπεται σε CALL 130K+100 χωρίς να τροποποιηθεί η ίδια η εντολή του προγράμματος.

45

45

Περιεχόμενα

- Βασικές αρχές διαχείρισης μνήμης.
- Διαμοίραση μνήμης.
- Σελιδοποίηση και κατάτμηση.

46

46

Βασικοί όροι

Πίνακας 7.1 Όροι Διαχείρισης Μνήμης

Όροι	Περιγραφή
Πλαίσιο (frame)	Ένα κομμάτι μνήμης *σταθερού* μεγέθους.
Σελίδα (page)	Ένα κομμάτι δεδομένων *σταθερού* μεγέθους στην περιφερειακή μνήμη (π.χ. δίσκος).
Τμήμα (segment)	Ένα κομμάτι δεδομένων *μεταβλητού* μεγέθους στην περιφερειακή μνήμη.

47

47

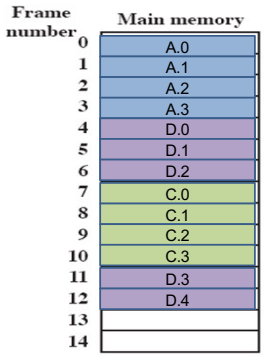
Σελιδοποίηση

- Η κύρια μνήμη χωρίζεται σε ισομεγέθη μικρά κομμάτια που λέγονται πλαίσια σελίδας (page frames).
- Κάθε πρόγραμμα χωρίζεται σε ένα αριθμό από σελίδες (όσες χρειασθούν ανάλογα με το πόσο μεγάλο είναι) οι οποίες έχουν ακριβώς το ίδιο μέγεθος με τα πλαίσια σελίδας της κύριας μνήμης.
- Όταν ένα πρόγραμμα φορτώνεται στην κύρια μνήμη, όλα τα μέρη του φορτώνονται σε αντίστοιχο αριθμό από πλαίσια.
- Για κάθε πρόγραμμα φορτωμένο στην κύρια μνήμη δημιουργείται ένας πίνακας σελίδων (page table) που δείχνει ποια πλαίσια καταλαμβάνονται από το πρόγραμμα.
- Τα πλαίσια αυτά δεν είναι απαραίτητο να είναι συνεχόμενα.
- Κάθε εγγραφή του πίνακα αποτελείται από μία ομάδα από bits πληροφοριών (π.χ. προστασίας, κλπ.) και ένα δείκτη που δείχνει τη θέση μνήμης από όπου ξεκινάει η αποθήκευση της σελίδας.
- Ένας καταχωρητής δείχνει τη θέση μνήμης από όπου ξεκινάει η αποθήκευση του πίνακα.

48

48

Παράδειγμα σελιδοποίησης



49

Πίνακας σελίδων για το προηγούμενο παράδειγμα

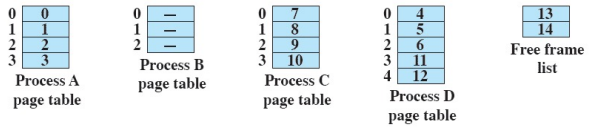


Figure 7.10 Data Structures for the Example of Figure 7.9 at Time Epoch (f)

50

Χαρακτηριστικά της σελιδοποίησης

- Το πλεονέκτημα της σελιδοποίησης είναι ότι μειώνει στο ελάχιστο δυνατόν τον κατακερματισμό της μνήμης: εξωτερικός κατακερματισμός δεν υπάρχει καθόλου ενώ ο εσωτερικός κατακερματισμός περιορίζεται στην τελευταία σελίδα ενός προγράμματος.
  - Αυτή η απλής μορφής σελιδοποίηση μπορεί να θεωρηθεί ότι είναι παραλλαγή της μεθόδου των σταθερών τμημάτων χωρίς τον περιορισμό να είναι τα τμήματα σε συνεχόμενες θέσεις μνήμης.
- Όμως για την αποκωδικοποίηση μίας διεύθυνσης μνήμης χρειάζονται δύο προσπελάσεις, μία για τον πίνακα σελίδων και μία για την ίδια τη σελίδα.
  - Το πρόβλημα αυτό μπορεί να επιλυθεί με τη χρήση *συσχετιστικών καταχωρητών* όπως θα δούμε παρακάτω.

51

Κατάτμηση

- Αν η σελιδοποίηση μπορεί να θεωρηθεί σαν παραλλαγή της μεθόδου των σταθερών τμημάτων χωρίς τον περιορισμό να είναι τα τμήματα φορτωμένα στη μνήμη σε συνεχόμενες θέσεις, η κατάτμηση (segmentation) μπορεί να θεωρηθεί η αντίστοιχη παραλλαγή της μεθόδου των μεταβλητών τμημάτων μνήμης.
- Οι δομές που δημιουργούνται είναι λίγο πολύ οι ίδιες με αυτές της σελιδοποίησης αλλά ο πίνακας τμημάτων πρέπει να έχει την επί πλέον πληροφορία του μεγέθους του τμήματος. Λόγω του διαφορετικού μεγέθους των τμημάτων δεν υπάρχει εσωτερικός κατακερματισμός αλλά παραμένει το πρόβλημα του εξωτερικού κατακερματισμού σε μικρότερο όμως βαθμό από ότι στη μέθοδο των μεταβλητών τμημάτων μνήμης.

52

Χαρακτηριστικά της κατάτμησης

- Η βασική ιδέα στην κατάτμηση είναι ότι ένα πρόγραμμα χωρίζεται συνήθως σε λογικές ενότητες (διαδικασίες, στοίβες, δομημένος προγραμματισμός, συνολική δήλωση των δομών δεδομένων σε συγκεκριμένα σημεία) και κάθε ενότητα αντιστοιχεί σε ένα τμήμα. Μάλιστα δε, είναι ο μεταφραστής αυτός που καθορίζει πώς ένα πρόγραμμα θα χωρισθεί σε τμήματα.
- Το μεγάλο πλεονέκτημα είναι ότι επειδή ένα τμήμα είναι τώρα μία λογική οντότητα (διαδικασία, δομή δεδομένων, ομάδα από μεταβλητές) γίνεται πιο εύκολος ο έλεγχος και η διαχείριση των τμημάτων.
  - Π.χ. αν επαναμεταγλωττισθεί μία μόνο διαδικασία ενός προγράμματος, τα τμήματα για το υπόλοιπο πρόγραμμα δεν χρειάζεται να αλλάξουν.
  - Επίσης κάποια τμήματα μπορεί να δηλωθούν μόνο για διάβασμα και κατ' επέκταση να διαμοιράζονται μεταξύ διαφόρων προγραμμάτων.
  - Σημειωτέον ότι ένα πρόγραμμα μπορεί να καταλάβει (μεταβλητού μεγέθους) τμήματα τα οποία να μην είναι σε συνεχόμενες θέσεις στη μνήμη.
- Όμως ο προγραμματιστής πρέπει να είναι ενήμερος της τεχνικής αυτής και να βοηθάει με την υιοθέτηση κατάλληλης μεθοδολογίας προγραμματισμού (δόμηση των δεδομένων και εντολών, χρήση διαδικασιών και αρθρωτού προγραμματισμού (modular programming), κλπ.)

53

Αναπαράσταση διευθύνσεων

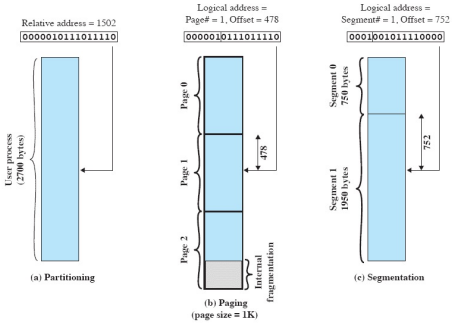
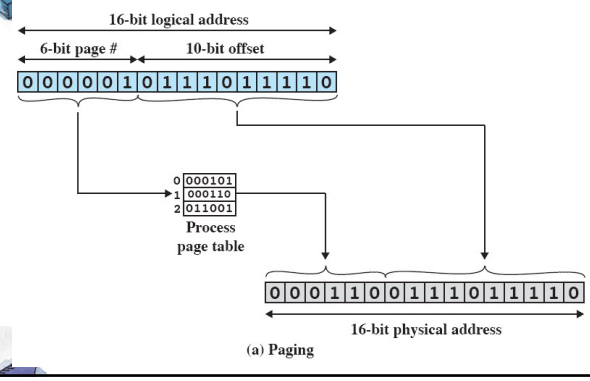


Figure 7.11 Logical Addresses

54

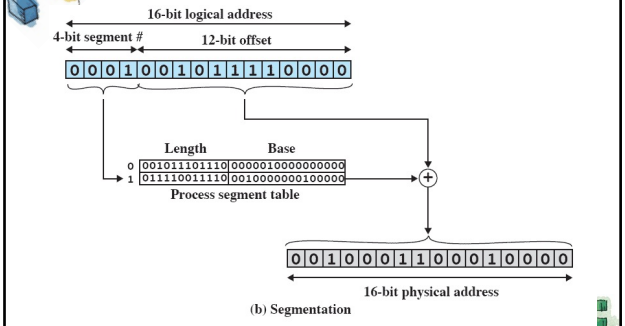
Μηχανισμός σελιδοποίησης



(a) Paging

55

Μηχανισμός κατάτμησης



(b) Segmentation

Figure 7.12 Examples of Logical-to-Physical Address Translation

56

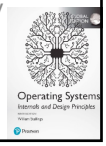
ΕΠΛ222: Λειτουργικά Συστήματα

(μετάφραση στα ελληνικά των διαφανειών του βιβλίου Operating Systems: Internals and Design Principles, 9/E, William Stallings)

Τέλος Ενότητας 7

Οι διαφάνειες αυτές έχουν συμπληρωματικό και επεξηγηματικό χαρακτήρα και σε καμία περίπτωση δεν υποκαθιστούν το βιβλίο

Γιώργος Α. Παπαδόπουλος
Τμήμα Πληροφορικής
Πανεπιστήμιο Κύπρου



57