



Περιέχει
δωρεάν CD-ROM

ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ

ΣΥΣΤΗΜΑΤΑ
ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΟΣ II

Ι. Κ. ΚΑΒΟΥΡΑ

Dip. Comp. Sci., M.Sc. (Comp. Sci.), Ph. D. (Comp. Sci.)



ΕΚΔΟΣΗ 7η
ΑΘΗΝΑ 2009

Περιεχόμενα

ΠΡΟΛΟΓΟΣ	xix
-----------------------	------------

ΚΕΦΑΛΑΙΟ 0: ΙΣΤΟΡΙΚΗ ΑΝΑΔΡΟΜΗ.....	1
---	----------

0.1. Συστήματα Χαρτοταινίας (Paper tape Systems)	1
0.2. Λειτουργικά Συστήματα Δίσκου (Disc Operating Systems, DOS)	3
0.3. Ακολουθιακά Συστήματα Επεξεργασίας Μιας Ροής (Single Stream Sequential Processing Systems).....	5
0.4. Συστήματα Δεσμίδων (Batch Systems).....	6
0.4.1. Βασικές λειτουργίες συστημάτων επεξεργασίας κατά δεσμίδες.....	8
0.4.2. Ενταμίευση και ψευδαπογραμμή	10
0.5. Συστήματα Πολυπρογραμματισμού (Multiprogramming Systems).....	12
0.6. Συστήματα Πραγματικού χρόνου (Real-time Systems)	14
0.7. Συστήματα Πολλαπλής πρόσβασης (Multi-access Systems)	16
0.8. Συστήματα Αλληλεπίδρασης (Interactive Systems) ή Διαλογικά Συστήματα (Dialog-Access Systems) ενός χρήστη	17
0.8.1. Βασικές ευκολίες διαλογικών συστημάτων	19
0.9. Παραθυρικά Λειτουργικά Συστήματα.....	20

ΚΕΦΑΛΑΙΟ 1: ΤΑΥΤΟΧΡΟΝΕΣ ΔΙΕΡΓΑΣΙΕΣ (CONCURRENT PROCESSES)	23
--	-----------

1.1. Εισαγωγή	23
1.2. Διεργασίες στη UCSD Pascal.....	27
1.3. Αμοιβαίος αποκλεισμός (Mutual exclusion)	30
1.4. Σηματοφορείς	37
ΑΣΚΗΣΕΙΣ 1.1.....	45

1.5. Συγχρονισμός Διεργασιών (Process Synchronization).....	43
1.6. Διεργασίες στο Σύστημα UNIX	50
1.6.1. Κλήσεις ελέγχου διεργασιών	52

ΚΕΦΑΛΑΙΟ 2: ΕΠΙΚΟΙΝΩΝΙΑ ΔΙΕΡΓΑΣΙΩΝ 59

2.1. Σύγχρονη και ασύγχρονη επικοινωνία	59
2.2. Πρόβλημα παραγωγού-καταναλωτή	61
ΑΣΚΗΣΕΙΣ 2.1.....	67
2.3. Παρακολουθητές (Monitors) ή Γραμματείς (Secretaries)	67
2.4. Μοντέλο Πελάτη-Εξυπηρετητή (Client-Server Model)	75
2.5. Επικοινωνία διεργασιών στην Ada.....	77
2.5.1. Επιλεκτική αναμονή (Selective waiting).....	81
2.6. Επικοινωνία διεργασιών με μηνύματα	84
ΑΣΚΗΣΕΙΣ 2.2.....	89
2.7. Επικοινωνία διεργασιών μέσω σωλήνων	90

ΚΕΦΑΛΑΙΟ 3: ΑΔΙΕΞΟΔΑ (DEADLOCKS)..... 93

3.1. Εισαγωγή.....	93
3.2. Αδιέξοδα με σειριακά επαναχρησιμοποιήσιμους πόρους	94
3.2.1. Έγκριση Αδιεξόδων (Deadlock Allowance)	96
3.2.2. Αυτόματη Ανακάλυψη και Αποκατάσταση Αδιεξόδων (Automatic Deadlock Detection and Recovery)	96
3.2.3. Πρόληψη Αδιεξόδων (Deadlock Prevention)	98
3.2.3.1 Ελεγχόμενη Καταχώριση (Controlled Allocation) ...	100
3.2.3.2 Διατεταγμένη Καταχώριση Πόρων (Standard Allocation Pattern ή Ordered Resource Policy).....	102
3.3. Αδιέξοδα με αναλώσιμους πόρους.....	103
3.3.1. Ιεραρχικά Συστήματα (Hierarchical Systems).....	104
3.4. Πρόβλημα Αναγνωστών και Συγγραφέων	106
ΑΣΚΗΣΕΙΣ 3.1.....	107

ΚΕΦΑΛΑΙΟ 4: ΔΟΜΗ ΕΝΟΣ ΥΠΟΘΕΤΙΚΟΥ ΣΥΣΤΗΜΑΤΟΣ..... 113

4.1.	Εισαγωγή.....	113
4.2.	Δομή επόπτη.....	117
4.2.1.	Συστήματα Μονολιθικού Παρακολουθητή (Monolithic Monitor Systems) ή Συγκεντρωτικά Συστήματα (Centralized Systems)	118
4.2.2.	Συστήματα Πολλών Παρακολουθητών (Multiple Monitor Systems) ή Αποκεντρωμένα Συστήματα (Decentralized Systems).....	119
4.2.2.1.	Διαστρωματικά Συστήματα (Layered Systems)	120
4.2.2.2.	Εικονικές Μηχανές (Virtual Machines).....	120
4.2.2.3.	Μέθοδος Πυρήνα (Nucleus Approach)	122
4.3.	Φιλοσοφίες Χρονοπρογραμματισμού (Scheduling Philosophies).....	124
4.4.	Κατάταξη χρονοπρογραμματιστών σ' επίπεδα	125
4.4.1.	Οργάνωση συστήματος εισόδου/εξόδου.....	127
4.4.2.	Οργάνωση συστήματος μνήμης.....	128
4.4.3.	Χρονικός προγραμματισμός ΚΜΕ.....	129
4.4.4.	Οργάνωση συστήματος αρχείων	131
4.4.5.	Περιβάλλον εκτέλεσης (run-time environment) των διεργασιών χρήση.....	132
4.5.	Το υποθετικό σύστημα	135
4.5.1.	Περιγραφή του συστήματος απ' τη βάση προς τα πάνω (bottom-up).....	137
4.6.	Δομή και διεπαφή του συστήματος UNIX	143
4.6.1.	Κλήσεις εισόδου/εξόδου	144
4.7.	Δομή της Αρχιτεκτονικής Windows NT	152

ΚΕΦΑΛΑΙΟ 5: ΠΥΡΗΝΑΣ (KERNEL) 157

5.1.	Εισαγωγή.....	157
5.2.	Οργάνωση διεργασιών.....	158
5.3.	Εικόνες διεργασιών.....	159
5.4.	Καταστάσεις διεργασιών.....	161

5.5.	Περιβάλλον διεργασιών.....	163
5.6.	Περιγραφητές διεργασιών	165
5.7.	Πίνακας περιγραφητών διεργασιών	165
5.8.	Ταυτότητες διεργασιών	166
5.9.	Υλοποίηση σηματοφορέων	166
5.9.1.	Υλοποίηση σηματοφορέων σε συστήματα ενός επεξεργαστή	167
5.9.2.	Υλοποίηση σηματοφορέων σε συστήματα πολυεπεξεργαστών	168
5.10.	Επικοινωνία διεργασιών μέσω γραμματοκιβωτίων	172
5.10.1.	Επικοινωνία διεργασιών στο υποθετικό σύστημα	176
5.11.	Ενεργοποίηση διεργασιών.....	180
5.12.	Επιστροφή από τις διαδικασίες εξυπηρέτησης διακοπών και παγίδων.....	181
5.13.	Διανομέας.....	182
5.14.	Εξυπηρέτηση διακοπών χρονομέτρου και λογιστική χρέωση.....	185
5.15.	Υλοποίηση διεργασιών στο σύστημα UNIX.....	189
5.15.1.	Οργάνωση και έλεγχος διεργασιών.....	189
5.15.2.	Δημιουργία και διαγραφή διεργασιών	192
5.15.3.	Συγχρονισμός διεργασιών μέσω σημάτων	194
5.15.4.	Μηχανισμοί επικοινωνίας	196
5.15.4.1.	Σωλήνες	197
5.15.4.2.	Ουρές μηνυμάτων	206

ΚΕΦΑΛΑΙΟ 6: ΔΙΑΧΕΙΡΙΣΗ ΕΙΣΟΔΟΥ/ΕΞΟΔΟΥ (I/O MANAGEMENT)..... 223

6.1.	Εισαγωγή.....	223
6.2.	Περιγραφητές συσκευών	228
6.3.	Εικονικές εντολές και υποπρογράμματα εισόδου/εξόδου	229
6.4.	Διαχειριστές περιφερειακών συσκευών	230
6.5.	Διαχειριστές τερματικών	232
6.6.	SPOOLers.....	235
6.7.	Κρυφή Μνήμη Δίσκου (Disk Cache)	237

6.8.	Πολιτικές διαχείρισης δίσκων	238
6.8.1.	Χρονικός προγραμματισμός δίσκων κινητής κεφαλής	238
6.8.2.	Χρονικός προγραμματισμός δίσκων σταθερής κεφαλής	242
6.9.	Διαχείριση Εισόδου/Εξόδου στο σύστημα UNIX	243
6.9.1.	Συσκευές ομάδων	244
6.9.2.	Συσκευές χαρακτήρων	246

ΚΕΦΑΛΑΙΟ 7: ΔΙΑΧΕΙΡΙΣΗ ΜΝΗΜΗΣ (MEMORY MANAGEMENT)..... 249

7.0.	Ιστορική αναδρομή.....	249
7.1.	Σκοποί.....	251
7.2.	Αναφορά διεύθυνσης μέσω Καταχωρητή Βάσης (Base Register Addressing)	252
7.3.	Καταχωρητές Βάσης (Base Registers) και Καταχωρητές Ορίου (Limit Registers).....	254
7.4.	Σελιδοποίηση (Paging)	256
7.4.1.	Μονάδα Διαχείρισης Μνήμης (Memory Management Unit, MMU) και Ενταμιευτής Παράπλευρης Μετάφρασης (Translation Lookaside Buffer, TLB)	262
7.5.	Τεμαχισμός (Segmentation)	263
7.6.	Τεμαχισμός με Σελιδοποίηση (Paged Segmentation).....	267
	ΑΣΚΗΣΕΙΣ 7.1.....	269
7.7.	Διαχείριση κρυφής μνήμης και αρχεία στο σύστημα MULTICS.....	273
7.8.	Διαχείριση μνήμης στα συστήματα της σειράς 68000 της MOTOROLA	275
7.9.	Διαχείριση μνήμης στα συστήματα Intel 80x86.....	278
7.9.1.	Τεμαχισμός.....	279
7.9.2.	Σελιδοποίηση	281

ΚΕΦΑΛΑΙΟ 8: ΔΙΑΧΕΙΡΙΣΗ ΜΝΗΜΗΣ (MEMORY MANAGEMENT)..... 283

8.1.	Εισαγωγή.....	283
8.2.	Καταστάσεις τεμαχίων	285

8.3.	Περιγραφητές τεμαχίων.....	286
8.3.1.	Λίστα κατειλημμένων (Occupied List) και Λίστα Ελεύθερων (Free List) περιοχών	287
8.4.	Πολιτικές προσκόμισης	288
8.4.1.	Πολιτικές Πρόβλεψης ή Προσδοκίας (Anticipatory Policies)....	289
8.4.2.	Πολιτικές Αίτησης (Demand Policies).....	290
8.5.	Πολιτικές τοποθέτησης.....	291
8.5.1.	Καλύτερο ταίριασμα (Best fit).....	291
8.5.2.	Χειρότερο ταίριασμα (Worst fit).....	292
8.5.3.	Πρώτο ταίριασμα (First fit).....	292
8.5.4.	Κυκλικό πρώτο ταίριασμα (Cyclic first fit)	293
8.5.5.	Γρήγορο ταίριασμα (Quick fit)	293
8.5.6.	Σύστημα Φίλων (Buddy System)	293
8.6.	Συμπύκνωση	295
8.7.	Πολιτικές αντικατάστασης	297
8.7.1.	Τυχαία επιλογή.....	301
8.7.2.	Γηραιότερο διαμένον (Oldest resident) ή Πρώτο-Μέσα-Πρώτο-Έξω (FIFO).....	301
8.7.3.	Λιγότερο Πρόσφατα Χρησιμοποιημένο (Least Recently Used, LRU)	301
8.7.3.1.	Υλοποίηση της στρατηγικής LRU με υλισμικό.....	302
8.7.3.2.	Υλοποίηση της στρατηγικής LRU με συνδυασμό υλισμικού και λογισμικού.....	303
8.7.4.	Λιγότερο Συχνά Χρησιμοποιημένο (Least Frequently Used, LFU)	304
8.7.5.	Δυφίο Χρήσης ή Αναφοράς (Use ή Reference Bit)	305
8.7.6.	Αλγόριθμος Δεύτερης Ευκαιρίας (Second Chance Algorithm)	306
8.7.7.	Προτεραιότητες Τεμαχίων	307
8.7.8.	Προτεραιότητες Διεργασιών	308
8.7.9.	Άλλες πολιτικές.....	309
8.7.10.	Τεμάχια εισόδου/εξόδου και (κατα)μεριζόμενα τεμάχια	311
	ΑΣΚΗΣΕΙΣ 8.1.	312
8.8.	Διαχείριση μνήμης στο σύστημα UNIX.....	314
8.8.1.	Διαχείριση μνήμης πριν από την έκδοση 3BSD	314
8.8.2.	Διαχείριση μνήμης στην έκδοση 4.2BSD	316

ΚΕΦΑΛΑΙΟ 9: ΔΙΑΧΕΙΡΙΣΗ ΚΜΕ (CPU MANAGEMENT)	321
9.1. Εισαγωγή	321
9.2. Στατική προτεραιότητα	323
9.3. Δυναμική προτεραιότητα.....	327
9.3.1. Πολιτικές που βασίζονται στη στάθμιση εισόδου/εξόδου και ΚΜΕ.....	331
9.3.2. Κυκλική Επαναφορά ή “Γύρω-Γύρω Όλοι” (Round Robin, RR)	332
9.3.3. Ουρές Πολλών επιπέδων (Multi-level Queues)	333
9.3.4. Χρονοπρογραμματισμός Προθεσμίας (Deadline Scheduling).....	338
9.3.5. Χρονοπρογραμματισμός οδηγούμενος από συναρτήσεις Πολιτικής (Policy-driven Scheduling)	340
9.4. Μέτρηση εξυπηρέτησης διεργασιών	342
9.5. Βαθμός πολυπρογραμματισμού.....	343
9.6. Έλεγχος Φόρτου (Load Control)	345
9.6.1. Έγκριση, ανακάλυψη και αποκατάσταση	346
9.6.1.1. Επέμβαση χειριστή	346
9.6.1.2. Λόγος εικονικού προς φυσικό χώρο	346
9.6.1.3. Χρόνος κύκλου	346
9.6.1.4. Συχνότητα σφαλμάτων	347
9.6.2. Πρόληψη ή αποφυγή	348
9.6.2.1. Προτεραιότητες διεργασιών	348
9.6.2.2. Σύνολο εργασίας.....	348
9.7. Προτεραιότητες νέων διεργασιών	350
9.8. Αδιέξοδα.....	351
9.8.1. Σειριακά επαναχρησιμοποιήσιμοι πόροι.....	352
9.8.2. Αναλώσιμοι πόροι.....	352
ΑΣΚΗΣΕΙΣ 9.1.....	353
9.9. Διαχείριση ΚΜΕ στο σύστημα UNIX.....	357
9.9.1. Έλεγχος φόρτου	359

ΚΕΦΑΛΑΙΟ 10: ΔΙΑΧΕΙΡΙΣΗ ΣΥΣΤΗΜΑΤΟΣ ΑΡΧΕΙΩΝ (FILE SYSTEM MANAGEMENT)	361
10.1. Εισαγωγή	361
10.2. Τύποι αρχείων	363
10.3. Μέθοδοι Προσπέλασης Αρχείων (File Access Methods) ή Λογική Οργάνωση Αρχείων (Logical File Organization)	364
10.3.1. Ακολουθιακή Προσπέλαση (Sequential Access)	364
10.3.2. Διαμερισμένη Προσπέλαση (Partitioned Access)	365
10.3.3. Δεικτοδοτημένη Ακολουθιακή Προσπέλαση (Indexed Sequential Access)	365
10.3.4. Άμεση Προσπέλαση (Direct Access)	367
10.3.5. Βασικές και μη βασικές μέθοδοι προσπέλασης	368
10.4. Ευρετήριο Συσκευής	370
10.5. Περιγραφητές Αρχείων	371
10.6. Ευρετήρια Αρχείων	373
10.6.1. Δομή ενός επιπέδου (single-level structure)	375
10.6.2. Δομή δύο επιπέδων (two-level structure)	375
10.6.3. Δεντροειδής δομή (Tree structure)	377
10.6.4. Δομή προσανατολισμένου ακυκλικού γράφου (DAG structure)	379
10.7. Μέθοδοι Αποθήκευσης Αρχείων (File Storage Methods) ή Φυσική Οργάνωση Αρχείων (Physical File Organization)	381
10.7.1. Διαχείριση ελεύθερου χώρου	382
10.7.2. Γειτονική καταχώριση (Contiguous allocation)	383
10.7.3. Συνδεδεμένη καταχώριση (Linked allocation)	386
10.7.4. Δεικτοδοτημένη καταχώριση (Indexed allocation)	388
10.8. Προστασία Αρχείων	390
10.9. Ακεραιότητα Συστήματος Αρχείων (File System Integrity)	393
10.10. Αξιοπιστία Συστήματος Αρχείων (File System Reliability)	397
ΑΣΚΗΣΕΙΣ 10.1	399
10.11. Σύστημα αρχείων του UNIX	402
10.11.1. Εισαγωγή	402
10.11.2. Λογική οργάνωση συστήματος αρχείων	403

10.11.3. Δομή και χρήση ευρετηρίου αρχείων.....	404
10.11.3.1. Ευρετήρια bin	406
10.11.3.2. Σύνδεσμοι	407
10.11.4. Φυσική οργάνωση συστήματος αρχείων.....	408
10.11.4.1. Ι-κόμβοι ή i-κόμβοι (Inodes ή i-nodes).....	410
10.11.4.2. Απεικόνιση διαδρομών (μονοπατιών) σε i-κόμβους	412
10.11.4.3. Απεικόνιση περιγραφητών αρχείων σε i-κόμβους	412
10.11.4.4. Πολιτικές καταχώρισης	415

ΚΕΦΑΛΑΙΟ 11: ΜΗΧΑΝΙΣΜΟΙ ΠΡΟΣΤΑΣΙΑΣ ΚΑΙ ΔΙΑΧΕΙΡΙΣΗΣ ΕΡΓΑΣΙΩΝ..... 419

11.1. Μηχανισμοί Προστασίας (Protection Mechanisms).....	419
11.2. Πεδία προστασίας (Protection Domains)	420
11.3. Ειδικοί μηχανισμοί προστασίας	422
11.3.1. Σύστημα UNIX	422
11.3.2. Σύστημα MULTICS.....	424
11.3.3. Intel 80x86	426
11.3.3.1. Ίδιο επίπεδο	428
11.3.3.2. Διαφορετικό επίπεδο.....	429
11.4. Γενικοί μηχανισμοί προστασίας.....	434
11.4.1. Λίστες Ελέγχου Προσπέλασης.....	434
11.4.2. Λίστες Ικανοτήτων.....	436
11.4.3. Μηχανισμός Κλειδιού/Κλειδαριάς.....	438
11.5. Δυναμικοί μηχανισμοί προστασίας	438
11.6. Διαχείριση Εργασιών (Job Management).....	441
11.7. Περιγραφητής εργασίας και ουρά εργασιών	442
11.8. Λογιστική εργασιών και έλεγχος πόρων	443
11.9. Δημιουργία διεργασιών	447
11.10. Διαγραφή διεργασιών	449
ΑΣΚΗΣΕΙΣ 11.1*	450

ΚΕΦΑΛΑΙΟ 12: ΕΙΣΑΓΩΓΗ ΣΤΑ ΣΥΣΤΗΜΑΤΑ ΠΡΑΓΜΑΤΙΚΟΥ ΧΡΟΝΟΥ ΚΑΙ ΣΤΗΝ ΑΞΙΟΠΙΣΤΙΑ 451

12.1. Εισαγωγή.....	451
12.2. Σύγχρονα και ασύγχρονα συστήματα.....	452
12.3. Δημοσκόπηση και διακοπές	454
12.4. Αλγόριθμοι χρονοπρογραμματισμού.....	456
12.5. Αξιοπιστία (Reliability).....	459
12.5.1. Αποφυγή βλαβών (fault avoidance)	461
12.5.1.1. Ελαχιστοποίηση βλαβών του υλισμικού	461
12.5.1.2. Ελαχιστοποίηση βλαβών του λογισμικού.....	462
12.5.1.3. Εντοπισμός σφαλμάτων (error detection).....	462
12.5.1.4. Θεραπεία βλαβών (fault treatment)	464
12.5.2. Επανόρθωση σφαλμάτων (error recovery).....	465
12.5.3. Ιεραρχική αντιμετώπιση σφαλμάτων	468

ΠΑΡΑΡΤΗΜΑ Α

ΟΙ ΚΥΡΙΟΤΕΡΕΣ ΕΝΤΟΛΕΣ ΤΩΝ ΛΕΙΤΟΥΡΓΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ ΕΝΟΣ ΧΡΗΣΤΗ MS-DOS, CP/M, UCSD P-SYSTEM ΚΑΙ ΟΙ ΑΝΤΙΣΤΟΙΧΕΣ ΕΝΤΟΛΕΣ ΤΟΥ UNIX..... 471

ΠΑΡΑΡΤΗΜΑ Β

ΚΥΡΙΟΤΕΡΕΣ ΚΛΗΣΕΙΣ ΤΟΥ ΣΥΣΤΗΜΑΤΟΣ UNIX..... 483

B.1. Έλεγχος διεργασιών	483
B.2. Διαχείριση μνήμης και προστασία	484
B.3. Επικοινωνία διεργασιών μέσω ουρών μηνυμάτων.....	484
B.4. Διαχείριση αρχείων και είσοδος/έξοδος	485
B.5. Διαχείριση ευρετηρίων	486
B.6. Διαχείριση χρόνων.....	486

ΠΑΡΑΡΤΗΜΑ Γ

ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΦΛΟΙΩΝ ΤΟΥ UNIX..... 487

Γ.1. Αναγνωριστικά μεταβλητών.....	488
Γ.2. Δηλώσεις, εντολές εκχώρησης και σχόλια	489

Γ.3. Εντολές και Διαδικασίες.....	492
Γ.4. Συνθήκες και Παραστάσεις	497
Γ.5. Υπό συνθήκη εντολές.....	499
Γ.5.1. Η εντολή if	500
Γ.5.2. Οι εντολές case και switch	501
Γ.6. Επαναληπτικές εντολές	502
Γ.6.1. Η εντολή while	502
Γ.6.2. Οι εντολές for και foreach	503
Γ.6.3. Οι εντολές until και repeat	505
Γ.7. Αντιμετώπιση παγίδων και διακοπών	506
Γ.8. Άλλες ευκολίες του φλοιού csh	508
Γ.8.1. Ιστορικός Μηχανισμός.....	508
Γ.8.2. Συνώνυμα ή Ψευδώνυμα.....	509
Γ.8.3. Έλεγχος Εργασιών (Job Control).....	510
ΑΣΚΗΣΕΙΣ Γ.1	513

ΠΑΡΑΡΤΗΜΑ Δ

ΔΙΑΧΕΙΡΙΣΗ ΕΡΓΑΣΙΩΝ ΣΤΑ ΣΥΣΤΗΜΑΤΑ ΔΕΣΜΙΔΩΝ 517

Δ.1. Εισαγωγή.....	517
Δ.2. Καταστάσεις εργασιών	519
Δ.3. Περιγραφητής εργασίας και ουρά εργασιών	520
Δ.4. Σχέση διαχειριστή εργασιών και spoolers	521
Δ.5. Κριτήρια διαχείρισης εργασιών.....	522
Δ.6. Πολιτικές προσανατολισμένες στη χρήση πόρων	525
Δ.7. Πολιτικές προσανατολισμένες στην εξυπηρέτηση.....	527
Δ.7.1. Πολιτική χειριστή.....	527
Δ.7.2. Πολιτική Εξυπηρέτησης με τη σειρά άφιξης (First Come First Served, FCFS – Πρώτη Αφιχθείσα, Πρώτη Εξυπηρετείται)	527
Δ.7.3. Πολιτική Μικρότερου Χρόνου Εξυπηρέτησης (Shortest Job First, SJF ή Shortest Job Next, SJN)	528
Δ.7.4. Πολιτική Μεγαλύτερου λόγου Απόκρισης (Highest Response ratio Next, HRN).....	529
Δ.7.5. Πολιτική Σταθερής Προτεραιότητας (Fixed Priority Policy) ...	530
Δ.7.6. Χρονοπρογραμματισμός Προθεσμίας (Deadline Scheduling) ..	530

Δ.8. Άλλες πολιτικές	532
Δ.9. Σχέση διαχειριστή εργασιών και διαχειριστή διεργασιών.....	533
ΑΠΑΝΤΗΣΕΙΣ ΑΣΚΗΣΕΩΝ	535
ΑΠΑΝΤΗΣΕΙΣ 1.1.	535
ΑΠΑΝΤΗΣΕΙΣ 2.1.	540
ΑΠΑΝΤΗΣΕΙΣ 2.2.	546
ΑΠΑΝΤΗΣΕΙΣ 3.1.	550
ΑΠΑΝΤΗΣΕΙΣ 6.1.	563
ΑΠΑΝΤΗΣΕΙΣ 7.1.	564
ΑΠΑΝΤΗΣΕΙΣ 8.1.	567
ΑΠΑΝΤΗΣΕΙΣ 9.1.	570
ΑΠΑΝΤΗΣΕΙΣ Γ.1.....	577
ΒΙΒΛΙΟΓΡΑΦΙΑ	589
Ξένη Βιβλιογραφία	589
Ελληνική Βιβλιογραφία.....	594
Άλλες Πηγές	595
ΕΥΡΕΤΗΡΙΟ	597

ΚΕΦΑΛΑΙΟ 7

ΔΙΑΧΕΙΡΙΣΗ ΜΝΗΜΗΣ (MEMORY MANAGEMENT)

ΜΕΡΟΣ ΠΡΩΤΟ: ΥΛΙΣΜΙΚΟ

7.0. Ιστορική αναδρομή

Οι μνήμες των *πρώτων υπολογιστών* είχαν μικρή χωρητικότητα και στοιχίζανε πολύ ακριβά. Οι προγραμματιστές των υπολογιστών αυτών ξόδευαν πολύ χρόνο προσπαθώντας να *επικαλύψουν (overlay)* τα προγράμματά τους στη μνήμη. Οι προγραμματιστές χώριζαν τα προγράμματά τους σε τμήματα που χωρούσαν στη διαθέσιμη μνήμη και που ήταν γνωστά ως *επικαλύψεις (overlays)*. Για να εκτελεστεί κάποιο πρόγραμμα, το πρώτο “*τμήμα επικάλυψης*” (*overlay segment*) φορτωνόταν στη μνήμη και εκτελούνταν για κάποιο χρονικό διάστημα. Όταν τελείωνε η εκτέλεσή του, διάβαζε και φόρτωνε στη μνήμη το δεύτερο τμήμα επικάλυψης και του μετέφερε τον έλεγχο κ.ο.κ., ώσπου να εκτελεστεί ολόκληρο το πρόγραμμα. Οι προγραμματιστές έπρεπε να αποφασίζουν οι ίδιοι για το πώς θα διαιρούσαν τα προγράμματά τους σε τμήματα, για το πού θα αποθηκεύονταν τα τμήματα αυτά στη βοηθητική μνήμη και ακόμη να φροντίζουν για τη μεταφορά τους από τη βοηθητική στην κύρια μνήμη (βλ. και Κεφ. 0.1).

Παρόλο που οι *πρώτοι σύγχρονοι μικροϋπολογιστές* διέθεταν σχετικά μεγάλη κεντρική μνήμη, μόνο ένα μέρος της μνήμης αυτής –της τάξης των 640 Kbytes– μπορούσε να προσπελαστεί άμεσα, λόγω του περιορισμένου μήκους λέξης των υπολογιστών αυτών. Έτσι τα λειτουργικά συστήματα των υπολογιστών αυτών επέτρεπαν –μέσω μιας ειδικής εντολής (που σε πολλά συστήματα, π.χ. συστήματα BASIC, ήταν γνωστή ως CHAIN (αλυσίδα)) ή μέσω των γλωσσών υψηλού επιπέδου– τη διαίρεση των προγραμμάτων σε **τμήματα (segments)**, δηλαδή σε σύνολα υποπρογραμμάτων, που μπορούσαν να φορτωθούν στη μνήμη ανεξάρτητα από το κύριο πρόγραμμα. Για παράδειγμα, τα υποπρογράμματα που δίνουν αρχικές τιμές

στις μεταβλητές ή στις δομές δεδομένων ενός προγράμματος, τα υποπρογράμματα που εμφανίζουν/εκτυπώνουν τα τελικά αποτελέσματα του προγράμματος και τα υποπρογράμματα που καλούνται (χρησιμοποιούνται) πολύ σπάνια κατά την εκτέλεση του προγράμματος μπορούν να δηλωθούν ως (ανεξάρτητα, ελπίζουμε) τμήματα.

Η δήλωση ενός υποπρογράμματος ως “τμήματος”¹ δεν επηρεάζει τη σημασιολογία του προγράμματος. Επηρεάζει μόνο το χρόνο εκτέλεσής του και τις απαιτήσεις του σε μνήμη. Το “τμήμα-υποπρόγραμμα” και όλα τα φωλιασμένα του υποπρογράμματα (εκτός αυτών που είναι επίσης τμήματα) αποτελούν μία ομάδα, που ονομάζεται *τμήμα κώδικα* (*code segment*). Ένα πρόγραμμα και τα υποπρογράμμά του μεταγλωττίζονται σε ένα τμήμα κώδικα, εκτός αν μερικά από τα υποπρογράμματά του έχουν δηλωθεί ως τμήματα.

Ένα τμήμα κώδικα *διαμένει στο δίσκο* (*disc resident*), μέχρις ότου να χρησιμοποιηθεί, και ο χώρος που καταλαμβάνει στη μνήμη μπορεί να επικαλυφθεί όταν τελειώσει η εκτέλεσή του. Έτσι η χρήση της διαθέσιμης μνήμης μπορεί να βελτιωθεί με το να δηλώσουμε τα υποπρογράμματα που χρησιμοποιούνται μόνο μία φορά ή σπάνια ως τμήματα. Η απροσδόκητη όμως συμπεριφορά των προγραμμάτων (για παράδειγμα, οι κλήσεις των υποπρογραμμάτων που εξαρτώνται από τα δεδομένα και η εκτέλεση των αναδρομικών υποπρογραμμάτων) καθιστά την εφαρμογή της τεχνικής αυτής αδύνατη σε πολλές περιπτώσεις.

Η εμφάνιση των *συστημάτων πολυπρογραμματισμού* στις αρχές της δεκαετίας του '60 σήμαινε ότι η μνήμη ενός υπολογιστή έπρεπε να μπορούσε να (κατα)μεριστεί συγχρόνως σε περισσότερα από ένα προγράμματα. Αν όλα τα προγράμματα μπορούσαν να φορτωθούν ταυτόχρονα, δε θα υπήρχε κανένα πρόβλημα. Στην πράξη όμως η χωρητικότητα της κύριας μνήμης είναι, ακόμα και στα σύγχρονα μεγάλα συστήματα, πεπερασμένη και τα προγράμματα των χρηστών δεν είναι πάντοτε μικρά.

¹ Στις UCSD, TURBO και THINK Pascal αυτό γίνεται με το να γραφεί η ειδική λέξη **segment** πριν από τις ειδικές λέξεις **process**, **procedure** ή **function**. Σ' ένα πρόγραμμα της UCSD Pascal μπορούν να δηλωθούν μέχρι 255 τμήματα. Τα σώματα των τμημάτων αυτών πρέπει να δηλωθούν πριν από τα σώματα των κοινών υποπρογραμμάτων που ανήκουν στο ίδιο τμήμα κώδικα. Έτσι, αν ένα τμήμα-υποπρόγραμμα χρειάζεται ένα υποπρόγραμμα που δεν είναι τμήμα, το τελευταίο πρέπει να δηλωθεί ως **forward**. Στη συμβολική γλώσσα των υπολογιστών της σειράς 86, τα τμήματα ορίζονται με τις ψευδοεντολές **SEGMENT** και **ENDS** (βλ. Τόμο Ι, Κεφ. 3.12.8.1).

Μία από τις πρώτες τεχνικές που χρησιμοποιούσαν τα συστήματα πολυπρογραμματισμού ήταν η **ανταλλαγή των προγραμμάτων (program swapping)** – γνωστή και ως **ανταλλαγή των εργασιών (job swapping)** ή ως *επαναφορά-εκτόπιση (roll-in-roll-out)*– μεταξύ της βοηθητικής και της κύριας μνήμης. Όταν ερχόταν η σειρά μιας (δι)εργασίας² να εκτελεστεί, τότε (το πρόγραμμα ή τα προγράμματά της) φορτωνόταν στη μνήμη και ενεργοποιούνταν. Όταν η (δι)εργασία εξαντλούσε το τεμάχιο (κβάντο) χρόνου που της είχε καταχωρηθεί ή όταν απαιτούσε είσοδο/έξοδο που την ανάγκαζε να περιμένει, τότε το πρόγραμμά της αποθηκευόταν πίσω στη βοηθητική μνήμη και φορτωνόταν το πρόγραμμα της επόμενης (δι)εργασίας κ.ο.κ. Αν υπήρχε αρκετή μνήμη, τότε μια (δι)εργασία μπορούσε να εκτελεστεί ταυτόχρονα με την ανταλλαγή των προγραμμάτων των άλλων (δι)εργασιών μεταξύ βοηθητικής και κύριας μνήμης.

Η τεχνική ανταλλαγής των προγραμμάτων έχει δύο *μειονεκτήματα*. Είναι, πρώτον, *χρονοβόρα*, αν και ο χρόνος επιβάρυνσης μπορεί να ελαττωθεί με τη χρήση επανεισαγόμενων προγραμμάτων και με τον περιορισμό των χρηστών σε μία μόνο γλώσσα υψηλού επιπέδου [όπως στα πρώτα “συστήματα (κατα)μερισμού χρόνου της BASIC” (BASIC time-sharing systems)].

Δεύτερον, τα προγράμματα εκτελούν συνήθως ένα υποσύνολο των υποπρογραμμάτων τους: τα υπόλοιπα υποπρογράμμά τους δεν είναι απαραίτητο να βρίσκονται διαρκώς στη μνήμη. Έτσι η ανταλλαγή ολόκληρων των προγραμμάτων μπορεί να είναι άσκοπη και να καταναλώνει περισσότερο χώρο της διαθέσιμης μνήμης απ’ όσο πραγματικά χρειάζονται τα προγράμματα αυτά.

7.1. Σκοποί

Οι κύριοι *σκοποί* διαχείρισης της μνήμης ενός συστήματος πολυπρογραμματισμού είναι οι εξής:

- (i) *Δυναμική μεταθεσιμότητα (Dynamic relocatability)*, που επιτρέπει τη γρήγορη τοποθέτηση των προγραμμάτων σε διαφορετικές θέσεις της μνήμης έτσι, ώστε να βελτιστοποιείται η χρήση της διαθέσιμης μνήμης.

² Στα πρώτα συστήματα πολυπρογραμματισμού χρησιμοποιούνταν ο όρος “εργασία” ή “πρόγραμμα” αντί του όρου “διεργασία”.

- (ii) *Προστασία (Protection)*, που εξασφαλίζει την ακεραιότητα των διεργασιών με το να τους απαγορεύεται η δυνατότητα ν' αλλάξουν τα περιεχόμενα των θέσεων της μνήμης που δεν ανήκουν στο χώρο των διευθύνσεών τους.
- (iii) *Εικονική ή λογική μνήμη (Virtual ή logical memory)* μεγάλης χωρητικότητας, που απαλλάσσει τους χρήστες από την περιορισμένη χωρητικότητα της πραγματικής (φυσικής) μνήμης.
- (iv) *Λογική οργάνωση του προγράμματος σε διάφορες δομοενότητες (modules)*, που μπορούν να μεταφραστούν, να προστατευτούν και να (κατα)μεριστούν ανεξάρτητα η μία από την άλλη.

Οι σκοποί αυτοί μπορούν να επιτευχθούν με τη βοήθεια του υλισμικού, όπως θα περιγράψουμε σ' αυτό το κεφάλαιο, και του σχετικού λογισμικού, που θα περιγράψουμε στο επόμενο κεφάλαιο.

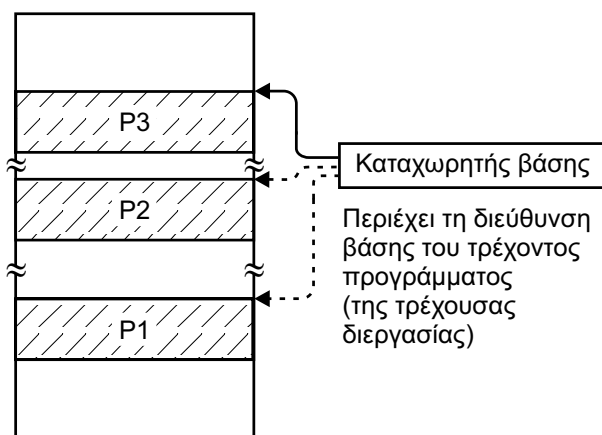
7.2. Αναφορά διεύθυνσης μέσω Καταχωρητή Βάσης (Base Register Addressing)

Η μέθοδος αναφοράς διεύθυνσης μέσω καταχωρητή βάσης είναι παρόμοια με τη σχετική αναφορά και την αναφορά διεύθυνσης μέσω καταχωρητή δείκτη, και έχει ήδη περιγραφεί στο Κεφάλαιο 3.5 του πρώτου τόμου. Η ΚΜΕ των υπολογιστών που χρησιμοποιούν τη μέθοδο αυτή περιέχει έναν καταχωρητή που ονομάζεται **καταχωρητής βάσης (base register)** ή **καταχωρητής μετάθεσης (relocation register)**. Στους υπολογιστές αυτούς η τελική διεύθυνση μιας εντολής αναφοράς στη μνήμη είναι ίση με το άθροισμα των περιεχομένων του καταχωρητή βάσης και της τελικής διεύθυνσης που προσδιορίζεται από την εντολή.

Η *λειτουργία του υλισμικού* των υπολογιστών αυτών διαφέρει από αυτή που περιγράψαμε στα Κεφάλαια 4.3 και 6.4.2 του πρώτου τόμου μόνο στο ότι οι εντολές εκχώρησης περιεχομένων καταχωρητών στον καταχωρητή διευθύνσεων της μνήμης, MAR, περιέχουν στο δεξί τους μέλος την πρόσθεση του όρου [Καταχωρητής_Βάσης].

Το *πλεονέκτημα* της μεθόδου αυτής είναι ότι επιτρέπει τη **δυναμική μετάθεση (dynamic relocation)** των προγραμμάτων. Σ' ένα μεγάλο σύστημα υπολογιστών συνυπάρχουν συνήθως ταυτόχρονα διάφορα προγράμματα στην κύρια μνήμη και είναι επιθυμητό να μπορούν να μετακινούνται αυτά εύκολα σ' οποιαδήποτε θέση

της διαθέσιμης μνήμης. Όλα τα προγράμματα μεταφράζονται σχετικά ως προς τη θέση 0 της μνήμης. Ένα ολόκληρο πρόγραμμα μπορεί να **φορτωθεί δυναμικά (dynamic loading)** σ' οποιαδήποτε διεύθυνση της μνήμης, με το να φορτωθούν οι εντολές και τα δεδομένα του στις θέσεις που αρχίζουν από τη διεύθυνση αυτή (*διεύθυνση βάσης*) και με το να αποθηκευτεί η διεύθυνση αυτή στον καταχωρητή βάσης, πριν εκτελεστεί (διανεμηθεί στην ΚΜΕ) το πρόγραμμα αυτό. Ο (*σχετικός*) **δυναμικός φορτωτής** δε χρειάζεται ν' αλλάξει τα πεδία διεύθυνσης των εντολών αναφοράς στη μνήμη. Σε πολλούς υπολογιστές η εντολή που φορτώνει τον καταχωρητή βάσης με μία διεύθυνση είναι προνομιούχα.



Σχήμα 7.1. Αναφορά διεύθυνσης μέσω καταχωρητή βάσης.

Οι προσωπικοί μικροϋπολογιστές της IBM, που βασίζονταν στους επεξεργαστές 8086 ή 80286, διέθεταν τέσσερις καταχωρητές βάσης που χρησιμοποιούνταν για τα τμήματα κώδικα, στοίβας, δεδομένων και “επιπλέον” δεδομένων όπως αναφέραμε στα Κεφάλαια 2.6.1 και 3.12.1 του πρώτου τόμου. Το *μειονέκτημα* αναφοράς διεύθυνσης μέσω καταχωρητών βάσης είναι η αδυναμία προστασίας των προγραμμάτων. Ένα πρόγραμμα μπορεί να προσπελάσει διαθέσιμη μνήμη που δεν του ανήκει και να καταστρέψει τα άλλα προγράμματα τα οποία διαμένουν ταυτόχρονα μ' αυτό στη μνήμη.

Ένας τρόπος που χρησιμοποιήθηκε στους υπολογιστές της σειράς 360 της IBM για να επιτευχθεί η προστασία της μνήμης ήταν ο εξής: Η μνήμη των υπολογιστών αυτών θεωρούνταν διαιρεμένη σε περιοχές μήκους 2K δυφιοσυλλαβών, καθεμία

από τις οποίες είχε έναν *κώδικα προστασίας (protection code)* τεσσάρων δυφίων. Η λέξη κατάστασης του προγράμματος (βλ. Τόμο I, Κεφ. 6.1) περιείχε ένα τετραψήφιο *κλειδί*. Το υλισμικό των υπολογιστών 360 παγίδευε κάθε προσπάθεια προσπέλασης μιας περιοχής της μνήμης, της οποίας ο κώδικας προστασίας διέφερε από το κλειδί που περιείχε η λέξη κατάστασης του προγράμματος της τρέχουσας διεργασίας. Εφόσον οι εντολές αλλαγής του κώδικα προστασίας και του κλειδιού ήταν προνομιούχες, μόνο το λειτουργικό σύστημα μπορούσε να τις χρησιμοποιήσει, κι έτσι οι διεργασίες των χρηστών και το λειτουργικό σύστημα μπορούσαν να προστατευτούν.

7.3. Καταχωρητές Βάσης (Base Registers) και Καταχωρητές Ορίου (Limit Registers)

Η ΚΜΕ κάποιων υπολογιστών περιέχει δύο καταχωρητές: έναν **καταχωρητή βάσης (base register)**, όπως περιγράψαμε παραπάνω, κι έναν **καταχωρητή ορίου (limit register)**, που περιέχει τη μέγιστη διεύθυνση του τμήματος στη μνήμη ή έναν **καταχωρητή μήκους (length register)**, που περιέχει το μήκος του τμήματος. Στους υπολογιστές αυτούς η τελική διεύθυνση των εντολών αναφοράς στη μνήμη είναι επίσης ίση με το άθροισμα των περιεχομένων του καταχωρητή βάσης και της τελικής αναφοράς που προσδιορίζουν οι εντολές.

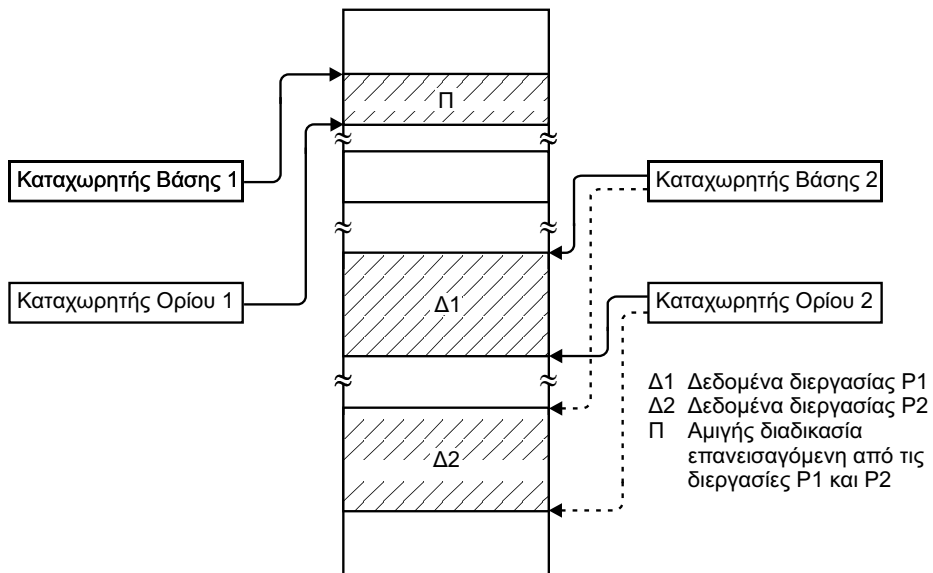
Η *δυναμική μετάθεση (dynamic relocation)* των προγραμμάτων επιτυγχάνεται επίσης με το να φορτωθούν οι εντολές και τα δεδομένα τους στις διευθύνσεις, που αρχίζουν από τη διεύθυνση βάσης, και με το να γίνει αποταμίευση της διεύθυνσης αυτής στον καταχωρητή βάσης του υπολογιστή. Η *προστασία* της μνήμης των διεργασιών επιτυγχάνεται, είτε αν συγκρίνουμε την τελική διεύθυνση των εντολών αναφοράς στη μνήμη με τα περιεχόμενα του καταχωρητή ορίου είτε αν συγκρίνουμε τα περιεχόμενα του πεδίου διεύθυνσης των εντολών αυτών με τα περιεχόμενα του καταχωρητή μήκους του υπολογιστή. Έτσι η *λειτουργία του υλισμικού* των υπολογιστών αυτών διαφέρει από αυτήν που περιγράψαμε στα Κεφάλαια 6.4.2 και 4.3 του πρώτου τόμου στο ότι περιλαμβάνει (μικρο)εντολές της μορφής:

```
if (τελική_διεύθυνση < 0 || [Καταχωρητής_Βάσης] +
    τελική_διεύθυνση) > [Καταχωρητής_Ορίου])
```

```

// ή (πεδίο_διεύθυνσης > [Καταχωρητής_Μήκους])
στείλε στο υλισμικό ένα σήμα
// το αντίστοιχο της εντολής signal
"παγίδας_παραβίασης_της_μνήμης";
// memory_violation_trap
else MAR = [Καταχωρητής_Βάσης] + τελική_διεύθυνση;

```



Σχήμα 7.2. Αναφορά διεύθυνσης μέσω καταχωρητών Βάσης και Ορίου.
[Περιεχόμενα των καταχωρητών Βάσης και Ορίου κατά την εκτέλεση της διεργασίας P1 (διεργασίας P2, διακεκομμένα)].

Το *μειονέκτημα* της μεθόδου αυτής είναι ότι ο **χώρος** των **εικονικών** ή **λογικών** ή **σχετικών διευθύνσεων** (**virtual** ή **logical** ή **relative address space**) του προγράμματος είναι αναγκαστικά μικρότερος από ή ίσος με τη χωρητικότητα της διαθέσιμης μνήμης, δηλαδή του **χώρου** των **πραγματικών** ή **φυσικών** ή **απόλυτων διευθύνσεων** (**real** ή **physical** ή **absolute address space**). Σε αντίθεση, αν ένας υπολογιστής διαθέτει δύο ζεύγη καταχωρητών βάσης και ορίου (ή μήκους), τότε μπορούν εύκολα να υλοποιηθούν επανεισαγόμενα προγράμματα (αμιγείς διαδικασίες), όπως δείχνει το Σχήμα 7.2.

Ο μεταγλωττιστής ή ο συμβολομεταφραστής ξεχωρίζει τις (αμετάβλητες) εντολές του προγράμματος από τα (μεταβλητά) δεδομένα του. Το τμήμα εντολών ή *τμήμα κώδικα (code segment)* του προγράμματος μπορεί έτσι να χρησιμοποιηθεί από διάφορες διεργασίες, εφόσον κάθε διεργασία έχει το δικό της *τμήμα δεδομένων (data segment)*. Οι προσωπικοί υπολογιστές που βασίζονταν στους επεξεργαστές 80286 διέθεταν τέσσερις καταχωρητές ορίου, ενώ αυτοί που βασίζονταν στους επεξεργαστές 80386 και πάνω είχαν έξι καταχωρητές (επιλογέων) τμημάτων και επιπλέον παρείχαν έναν ιεραρχικό μηχανισμό προστασίας όπως θα περιγράψουμε στα Κεφάλαια 7.9.1 και 11.3.3.

7.4. Σελιδοποίηση (Paging)

Η κύρια μνήμη (ο χώρος των φυσικών διευθύνσεων) των υπολογιστών που χρησιμοποιούν σελιδοποίηση αποτελείται από ισομεγέθη τμήματα ορισμένου μήκους –γνωστού ως *μεγέθους σελίδας (page size)*– που ονομάζονται **πλαίσια σελίδων (page frames)** και κάθε πρόγραμμα (ο χώρος των εικονικών διευθύνσεων) θεωρείται ότι είναι διαιρεμένο(ς) σε **σελίδες (pages)** του ίδιου μεγέθους. Τα πλαίσια σελίδων μπορούν να κατανεμηθούν (καταχωρηθούν) στις διεργασίες που βρίσκονται στο σύστημα έτσι, ώστε σε κάποια χρονική στιγμή μια διεργασία να έχει μερικές από τις σελίδες της –τις *ενεργές της σελίδες (active pages)*– στην κύρια μνήμη και τις υπόλοιπες –τις *μη ενεργές της σελίδες (inactive pages)*– στη βοηθητική μνήμη.

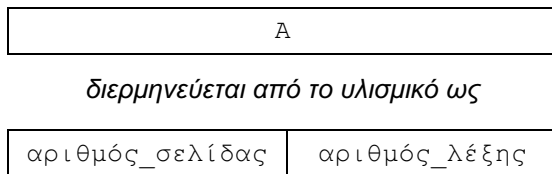
Για να υλοποιηθεί η σελιδοποίηση, χρειάζεται ένας *πίνακας σελίδων (page table)* για κάθε διεργασία. Ο πίνακας αυτός (ή η βάση του) αποθηκεύεται στο περιβάλλον της διεργασίας όπως περιγράψαμε στο Κεφάλαιο 5.5. Κάθε στοιχείο του πίνακα αυτού ονομάζεται **περιγραφητής της αντίστοιχης σελίδας (page descriptor)** του προγράμματος και περιέχει τα εξής:

- (i) ένα **δυφίο παρούσας (present bit)**, που καθορίζει αν η σελίδα βρίσκεται στην κύρια μνήμη ή όχι
- (ii) τη διεύθυνση (ή τον αριθμό του πλαισίου) της σελίδας στη βοηθητική ή/και στην κύρια μνήμη αντίστοιχα, και πιθανώς

(iii) **δυφία** που καθορίζουν τον τρόπο **προστασίας (protection bits)** της σελίδας³ και ίσως κι άλλες λεπτομέρειες.

Επιπλέον, το υλισμικό ενός συστήματος που χρησιμοποιεί σελιδοποίηση πρέπει να διαθέτει έναν ειδικό καταχωρητή, που ονομάζεται **καταχωρητής πίνακα σελίδων (page table register)** και περιέχει τη διεύθυνση βάσης του πίνακα των σελίδων της *τρέχουσας διεργασίας* πριν αυτή διανεμηθεί. Για να προσδιοριστεί η τελική (φυσική) διεύθυνση της μνήμης στην οποία αναφέρεται μία εντολή (αναφοράς στη μνήμη), το υλισμικό διερμηνεύει τις (εικονικές) διευθύνσεις των εντολών του προγράμματος με τέτοιο τρόπο, ώστε τα πλέον σημαντικά δυφία του πεδίου διεύθυνσης της εντολής να αποτελούν τον *αριθμό της σελίδας (page number)* του προγράμματος, ενώ τα λιγότερο σημαντικά δυφία να αποτελούν τον *αριθμό της λέξης (word number)* μέσα στη σελίδα αυτή.

Έτσι η τελική διεύθυνση μιας εντολής αναφοράς στη μνήμη



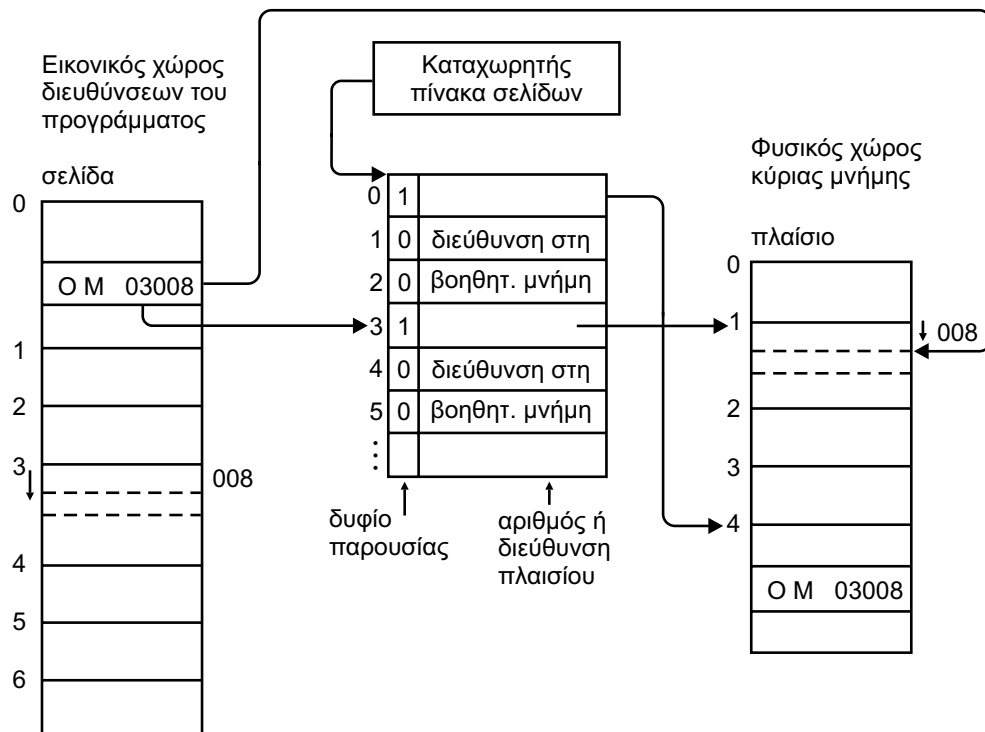
όπου ο αριθμός_σελίδας αποτελεί τα πλέον σημαντικά δυφία της τελικής διεύθυνσης και ο αριθμός_λέξης τα λιγότερο σημαντικά δυφία αυτής⁴.

Εννοείται ότι το μήκος του πεδίου διεύθυνσης των εντολών αναφοράς στη μνήμη είναι αρκετό για να μπορούν οι εντολές αυτές να αναφερθούν σε *ολόκληρη την εικονική μνήμη* (που είναι σχεδόν πάντα πολύ μεγαλύτερη από τη φυσική μνήμη). Ο αριθμός της σελίδας χρησιμοποιείται από το υλισμικό, για να επιλέξει το αντίστοιχο στοιχείο του πίνακα των σελίδων της διεργασίας. Αν το δυφίο παρουσίας υποδεικνύει ότι η σελίδα βρίσκεται στην κύρια μνήμη (είναι

³ Σε μερικά συστήματα τα δυφία αυτά μπορεί απλώς να καθορίζουν αν η σελίδα αποτελείται από εντολές (*code page*) ή από δεδομένα (*data page*), ενώ σ' άλλα συστήματα αν η σελίδα επιτρέπεται να προσπελαστεί για ανάγνωση μόνο (*Read-Only access*), για εγγραφή (*Write access*), για τροποποίηση (*Read-Write access*) ή για εκτέλεση (*Execute access*).

⁴ Οι αριθμοί αυτοί είναι ίσοι με $A/\text{μέγεθος_σελίδας}$ και $A\% \text{μέγεθος_σελίδας}$, αντίστοιχα, διότι το μέγεθος_σελίδας είναι μία δύναμη του δύο, για παράδειγμα 64, 128, 256 ή 512 λέξεις κ.λπ. έως 4 Kbytes ή και παραπάνω.

ίσο με 1, true), τότε ο περιγραφητής σελίδας προσδιορίζει το πλαίσιο της μνήμης που περιέχει τη σελίδα.



Σχήμα 7.3. Αναφορά διεύθυνσης μέσω περιγραφητών σελίδων.

Ο αριθμός της λέξης προστίθεται στη διεύθυνση του πλαισίου αυτού για να δώσει την πραγματική (φυσική) τελική διεύθυνση. Η **απεικόνιση της διεύθυνσης (address mapping)** από εικονική σε φυσική επιτυγχάνεται αυτόματα από το υλισμικό για κάθε αναφορά κάθε διεργασίας στην (εικονική και τελικά φυσική) μνήμη όπως δείχνει το Σχήμα 7.3. Αν η απεικόνιση της διεύθυνσης δεν είναι δυνατή, διότι το δυσφίο παρουσίας υποδεικνύει ότι η σελίδα δε βρίσκεται στην κύρια μνήμη, τότε το υλισμικό προξενεί μία διακοπή, που είναι γνωστή ως **σφάλμα σελίδας (page fault)**.

Η *λειτουργία του υλισμικού* ενός συστήματος που χρησιμοποιεί σελιδοποίηση διαφέρει από αυτήν που περιγράψαμε στα Κεφάλαια 6.4.2 και 4.3 του πρώτου τόμου στο ότι περιλαμβάνει (μικρο)εντολές της μορφής:


```

αριθμός_σελίδας = τελική_διεύθυνση.αριθμός_σελίδας και,
παράλληλα, αριθμός_λέξης =
    τελική_διεύθυνση.αριθμός_λέξης;
/* εφόσον το μέγεθος της σελίδας είναι κάποια δύναμη
του 2, απομονώνονται από το υλισμικό απλώς τα πεδία
αριθμός_σελίδας και αριθμός_λέξης */
if (πίνακας_σελίδων [αριθμός_σελίδας].δυψίο_παρουσίας)
    MAR = πίνακας_σελίδων [αριθμός_σελίδας].
        διεύθυνση_πλαισίου5 + αριθμός_λέξης;
else στείλε σήμα "διακοπής_σφάλματος_σελίδας";

```

Η διακοπή σφάλματος σελίδας (που εξυπηρετείται από την αντίστοιχη διαδικασία του πυρήνα) αφυπνίζει το διαχειριστή της μνήμης (βλ. Κεφ. 4.4.1 και 8.1), που προσκομίζει τη ζητούμενη σελίδα από τη βοηθητική μνήμη και την τοποθετεί στην κύρια μνήμη ενημερώνοντας τον πίνακα σελίδων της διεργασίας έτσι, ώστε να μπορέσει να προχωρήσει η εκτέλεσή της. Ο διαχειριστής της μνήμης (λογισμικό) πρέπει επίσης να διαλέξει, όταν συμβεί η διακοπή, ποια σελίδα θα πρέπει να αντικαταστήσει με τη ζητούμενη, αν δεν υπάρχει κενό πλαίσιο στη μνήμη όπως θα περιγράψουμε στο επόμενο κεφάλαιο.

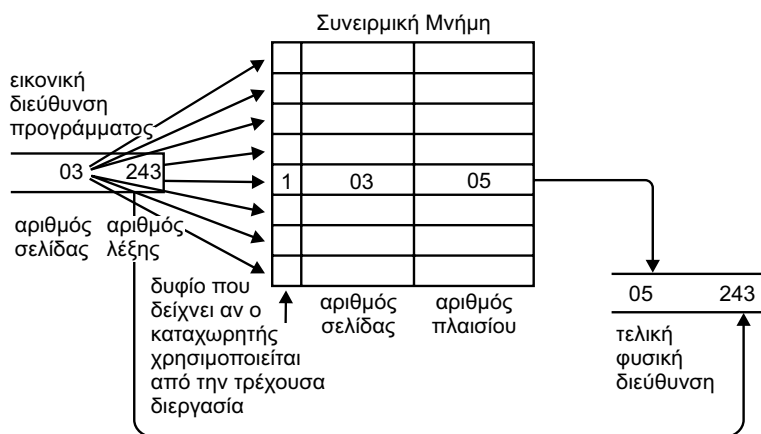
Παρατηρούμε ότι με τον παραπάνω μηχανισμό χρειάζονται δύο προσπελάσεις για κάθε αναφορά στη μνήμη: μία προσπέλαση στον πίνακα των σελίδων και μία στην τελική διεύθυνση. Για να αποφευχθεί η πρώτη προσπέλαση, χρησιμοποιούνται συνήθως (8 έως 16)⁶ γρήγοροι **συνειρμικοί καταχωρητές** ή **καταχωρητές συσχέτισης (associative registers)**, που περιέχουν αντίγραφα των περιγραφητών των σελίδων στις οποίες έγιναν οι πιο πρόσφατες αναφορές.

Όταν ανακληθεί μια εντολή του προγράμματος που αναφέρεται στη μνήμη, ο αριθμός της σελίδας της συγκρίνεται αυτόματα από το υλισμικό ταυτόχρονα με τους αριθμούς των σελίδων που περιέχονται σ' όλους τους συνειρμικούς καταχωρητές (απ' όπου προήλθε και ο όρος *συνειρμική μνήμη* ή *αποθήκη (associative memory* ή *store*, βλ. Τόμο I, Κεφ. 1.1.10). Επειδή οι περισσότερες αναφορές γίνονται στις σελίδες

⁵ Ή αριθμός_πλαισίου*μέγεθος_σελίδας (ο πολλαπλασιασμός υλοποιείται με προς τ' αριστερά μετατόπιση).

⁶ Ο Denning έδειξε ότι 8 ως 16 συνειρμικοί καταχωρητές είναι αρκετοί για να εκτελείται το πρόγραμμα χωρίς αισθητή επιβράδυνση (βλ. Άσκ. και Απάντ. 7.1.8).

που έχουν πιο πρόσφατα προσπελαστεί, υπάρχει μεγάλη πιθανότητα να βρεθεί ο αριθμός της σελίδας και να απομονωθεί από ένα συνειρμικό καταχωρητή ο αριθμός ή η διεύθυνση του πλαισίου που την περιέχει όπως δείχνει το Σχήμα 7.4.



Σχήμα 7.4. Αναφορά διεύθυνσης μέσω συνειρμικού καταχωρητή.

Αν ο αριθμός της σελίδας δε βρεθεί, τότε το υλισμικό αναφέρεται στον πίνακα των σελίδων της διεργασίας για να εντοπίσει αν η σελίδα βρίσκεται στη μνήμη ή όχι. Αν η σελίδα είναι παρούσα στη μνήμη, τότε απλώς φορτώνεται ο περιγραφητής της σ' έναν από τους συνειρμικούς καταχωρητές (αφού προηγουμένως αποταμιευτούν τα περιεχόμενα αυτού του καταχωρητή στον αντίστοιχο περιγραφητή)⁷. Οι καταχωρητές φορτώνονται συνήθως κυκλικά (cyclic).

Αν, όμως, η σελίδα δε βρίσκεται στην κύρια μνήμη, τότε το υλισμικό προκαλεί μία διακοπή και το λογισμικό ακολουθεί τη διαδικασία που αναφέραμε παραπάνω, για να προσκομίσει και να τοποθετήσει τη σελίδα. Το λογισμικό χρειάζεται επίσης να ενημερώσει τη συνειρμική μνήμη, αποθηκεύοντας τον περιγραφητή της σελίδας

⁷ Εννοείται ότι υπάρχουν περισσότερα πλαίσια σελίδων από καταχωρητές συσχέτισης. Στον πρώτο υπολογιστή που χρησιμοποίησε σελιδοποίηση, στον ATLAS (βλ. Τόμο Ι, Κεφ. 1.1.12), ο οποίος είχε (φυσική) μνήμη 16K λέξεων, υπήρχαν 32 συνειρμικοί καταχωρητές, δηλαδή τόσοι όσοι τα πλαίσια σελίδων που είχε η μνήμη του. Κάθε σελίδα περιείχε 512 (2^9) λέξεις. Το πεδίο διεύθυνσης των εντολών αναφοράς στη μνήμη είχε μήκος 20 δωφίων, δίνοντας έτσι εικονική μνήμη χωρητικότητας 2^{20} λέξεων. Τα 11 (πλέον σημαντικά) δωφία παρίσταναν τον αριθμό της σελίδας, ενώ τα υπόλοιπα 9 (λιγότερο σημαντικά) τον αριθμό της λέξης.

που αντικαθιστά (στον πίνακα των σελίδων της) και φορτώνοντας τον περιγραφητή της ζητούμενης σελίδας (από τον πίνακα των σελίδων της) στον αντίστοιχο συνειρμικό καταχωρητή.

Η λειτουργία του υλισμικού των υπολογιστών που χρησιμοποιούν συνειρμική μνήμη περιέχει (μικρο)εντολές της μορφής:

```
αριθμός_σελίδας = τελική_διεύθυνση.αριθμός_σελίδας και,
παράλληλα, αριθμός_λέξης =
                                τελική_διεύθυνση.αριθμός_λέξης;
                                // όπως περιγράψαμε παραπάνω
εντόπισε_παράλληλα (συνειρμική μνήμη, αριθμός_σελίδας,
                                διεύθυνση8, βρέθηκε);
if (!βρέθηκε)
if (!πίνακας_σελίδων[αριθμός_σελίδας].δυψίο_παρουσίας)
                                σήμανε διακοπή_σφάλματος_σελίδας;
else
{
    αποταμίευσε τα περιεχόμενα του (επόμενου)
                                συνειρμικού καταχωρητή;
    φόρτωσε τον καταχωρητή αυτόν με το ζεύγος
    (αριθμός_σελίδας, πίνακας_σελίδων[αριθμός_σελίδας].
                                διεύθυνση_πλαισίου);
    MAR = καταχωρητής_συσχέτισης.διεύθυνση9 +
                                αριθμός_λέξης;
}
else MAR = καταχωρητής_συσχέτισης.διεύθυνση +
                                αριθμός_λέξης;
```

Κάθε φορά που διανέμεται η ΚΜΕ σε μία νέα διεργασία (βλ. Κεφ. 5.1), τα περιεχόμενα της συνειρμικής μνήμης πρέπει να διαγραφούν, διότι η νέα διεργασία μπορεί ν' αναφέρεται στους ίδιους αριθμούς σελίδων με την προηγούμενη. Επειδή η διανομή της ΚΜΕ πρέπει να γίνεται όσο το δυνατόν ταχύτερα, αν η διαγραφή

⁸ Η αριθμός_πλαισίου.

⁹ Η αριθμός_πλαισίου*μέγεθος_σελίδας (ο πολλαπλασιασμός υλοποιείται με μετατόπιση προς τ' αριστερά).

ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ

ΣΥΣΤΗΜΑΤΑ ΥΠΟΛΟΓΙΣΤΩΝ ΤΟΜΟΣ II

Το βιβλίο αυτό καλύπτει:

- Ιστορική αναδρομή των λειτουργικών συστημάτων
- Παράλληλο προγραμματισμό
- Ταυτόχρονες διεργασίες
- Επικοινωνία διεργασιών
- Αδιέξοδα
- Δομή λειτουργικών συστημάτων και πυρήνων τους
- Διαχείριση εισόδου/εξόδου
- Διαχείριση μνήμης
- Χρονοπρογραμματισμό της Κεντρικής Μονάδας Επεξεργασίας
- Συστήματα αρχείων
- Μηχανισμούς προστασίας και διαχείρισης εργασιών
- Εισαγωγή στα συστήματα πραγματικού χρόνου και την αξιοπιστία

Περιέχει επίσης λεπτομερή τεχνική περιγραφή του λειτουργικού συστήματος UNIX, καθώς και πολλές λυμένες ασκήσεις σε TURBO Pascal, C και C-shell.

Ο **Γιάννης Κάβουρας** είναι Καθηγητής στο Τμήμα Πληροφορικής του Οικονομικού Πανεπιστημίου Αθηνών (ΟΠΑ). Σπούδασε Φυσικές Επιστήμες στο Πανεπιστήμιο Αθηνών, και κατέχει τους μεταπτυχιακούς τίτλους Diploma in Computing Science (University of Glasgow), Master of Science (University of London, ICS) και Ph.D. (University of Glasgow) στην Επιστήμη Υπολογιστών (Computer Science). Έχει εργαστεί ως "Lecturer" στο Glasgow College of Technology (τώρα Glasgow Caledonian University), στο Heriot-Watt University στο Εδιμβούργο και στο University of Glasgow, καθώς και ως "Επιστήμων Ερευνητής" στο Ε.ΚΕ.Φ.Ε. "Δημόκριτος". Διατέλεσε Επίτιμος (Αμισθος) Πρόξενος της Ελλάδος στη Γλασκόβη και στην περιοχή Strathclyde της Σκωτίας. Προϊστάμενος του Κέντρου Υπολογιστών του ΟΠΑ, επανειλημμένως Πρόεδρος του Τμήματος Πληροφορικής του Πανεπιστημίου αυτού και Πρόεδρος του Γνωμοδοτικού Τεχνικού Συμβουλίου του Υπουργείου Εσωτερικών, Δημόσιας Διοίκησης και Αποκέντρωσης. Υπήρξε κριτής επιστημονικών άρθρων για την Association of Computer Machinery (ACM) και άλλους ερευνητικούς φορείς, μέλος της ACM, μέλος και "Fellow" της British Computer Society (BCS), και μέλος της ΕΠΥ. Έχει, μεταξύ άλλων, γράψει πανεπιστημιακά συγγράμματα στους τομείς του Προγραμματισμού και της Οργάνωσης Υπολογιστών, καθώς και των Λειτουργικών και Κατανεμημένων Συστημάτων, οι οποίοι περιλαμβάνονται στα επιστημονικά του ενδιαφέροντα.

Επισκεφθείτε μας στο Internet:
www.klidarithmos.gr



ΕΚΔΟΣΕΙΣ
ΚΛΕΙΔΑΡΙΘΜΟΣ

Λορικού 4, Στάθμος Ληρίδας, 10440 ΑΘΗΝΑ, Τηλ. 210-5237635

ISBN 978-960-461-244-4



9 789604 612444