

# Index

- Abstract syntax tree, 64–65
- Access control list (ACL), 594–595
- Access path, 566
  - definition, 566
- Activation record, 169
- Ada
  - concurrent programming, 437–443
  - package, 437
  - real time programming, 439–443
  - task, 437–439
- Address translation unit (ATU), 480, 483, 485, 489, 492, 493, 511
- Allocation data structures, 37, 52–57, *see* *also* Stacks, Heaps, Search data structures
- Application domain, 1
- Array allocation, 177–179
- Assembler, 86–130
  - analysis phase, 92–93
  - data structures, 99, 125, 127
  - intermediate code, 94, 101–104, 110
  - pass structure, 94–95, 104
  - single pass assembler, 111, 121–130
  - synthesis phase, 92
  - two pass assembler, 95–111
- Assembly language, 86–91, *see also* Intel 8088 assembly language
  - advantages, 90–91
  - declaration statements, 89–90
  - directives, 90, 96–98
  - imperative statements, 89
  - literal, 90
- Atomic action, 581–583
- Authentication, 592
- Backtracking, 67
- Batch processing, 277–287
  - memory management, 280
  - sharing and protection, 280–286
- Binary program, 226
- Binding, 29–31
  - definition, 29
- Binding time, 29–31
- Blocking factor, 549–550
  - definition, 549
- Blocking of records, 548–550
- BNF, 24
- Bottom up parsing, 75–84
  - LALR parsing, 83–84
  - operator precedence parsing, 80–83
- Buddy system, 465–467
- Buffering of records, 545–548
- Busy wait, 402
- Capability
  - based addressing, 598–599
  - based protection, 596–602
  - definition, 596
  - in software, 601–602
  - structure, 597
- Capability list (C-list), 595–596
- Command interpreter, 281–285
- Command language, 265
- Command menu, 265
- Communication protocol, 615–619
- Compiler
  - back end, 16–19
  - basic block, 203
  - calling convention, 195
  - code generation, 17–18
  - code generator, 180–186
  - code optimization, 199–218
  - control flow analysis, 207
  - control structure, 192
  - control structure compilation, 192–198
  - data flow analysis, 208–211
  - display, 173

- dope vector, 178
- dynamic memory allocation, 166, 168–176
- dynamic pointer, 170–171
- expression compilation, 180–192
- front end, 13–16
- global optimization, 203, 206–211
- intermediate code, 14, *see also* Intermediate code
- local optimization, 203–206
- memory allocation, 17, 165–180
- memory binding, 166
- operand descriptor, 181
- parameter passing, 196–198
- register descriptor, 182
- static memory allocation, 166
- static pointer, 172
- tables, 14
- Compiler writing tool, *see* LEX, YACC
- Condition variable, 430–432
- Conditional critical region (CCR), 422–425
- Contiguous memory allocation, 471–478
- Control structure, 165
- Critical region (CR), 419–422
- Critical section (CS), 399–408
  - algorithmic implementation, 402–407
  - definition, 399
  - properties of implementation, 399–400
- Data structures, 36–57
  - Linear data structure, 36
  - Nonlinear data structure, 36
- Deadlock, 371–395
  - conditions for, 376–377
  - definition, 371
- Deadlock handling, 377–395
  - Banker's algorithm, 389–393
  - deadlock avoidance, 386–393
  - deadlock characterization, 377–382
  - deadlock detection, 383–385
  - deadlock prevention, 386–388
  - deadlock resolution, 385–386
  - mixed approach, 393–394
- Debug monitor, 260–261
- Degree of multiprogramming, 302
- Derivation, 21–22
- Detranslator, 3
- Device driver, 540–542
- Dining philosophers problem, 410–411
- Direct manipulation system, 265
- Directory, 563–568
  - current directory, 565
  - hierarchy, 564–565
  - home directory, 565
  - mounting, 567–568
- Disk mirroring, 581
- Disk space allocation, 569–571
  - indexed allocation, 570–571
  - linked allocation, 570
- Distributed control algorithm, 624–633
  - deadlock handling, 629–633
  - mutual exclusion, 625–629
- Distributed operating system, 604–649
  - design issues, 608–610
  - file system, 633–637
  - reliability, 637–642
  - resource allocation, 622–624
  - security, 643–649
- Distributed system, *see also* Event precedence, Distributed control algorithm
  - definition, 605
  - model, 606
  - state, 619–620
- Dynamic binding, 30
- Editor, 257–259
- Encryption, 588–591, 643–646
  - definition, 589
  - one-way function, 589
  - private key encryption, 644
  - public key encryption, 645
- EQU statement, 96, 117
- Event, 324
- Event control block (ECB), 355
- Event monitoring, 354–357
- Event precedence, 620–622
- Execution domain, 1
- Execution gap, 2
- Execution profile, 255
- Fault tolerance, 580–583, 639–641
- File access, 571–574
- File control block (FCB), 562

- File label, 562
- File map table (FMT), 570
- File organization, 542–544
- File protection, 569, 592–596
- File sharing, 576–577
- File system, 521, 561–586, *see also* File access, File sharing, File protection, Atomic action, Disk mirroring
  - actions at file close, 574
  - actions at file open, 572–573
  - integrity, 578–580
  - reliability, 578–584, *see also* Recovery, Fault tolerance
- Finite state automaton (FSA), 60
  - Deterministic FSA (DFA), 60
- Fork–join, 332–333
- Formal language, 19
- Forward reference, 11
- Garbage collection, 475–476
- Hash table
  - collision, 44
  - collision handling, 46–49
  - hashing function, 44–46
- Hash tables, 44–49
- Heap, 55–57, 461–464
- Indivisible operation, 413
- Intel 8088
  - architecture, 111–113
  - assembly language, 117–121
  - instructions, 113–117
  - linker, 240–243
    - autolinking, 239
  - object module, 233–239
  - single pass assembler, 111, 121–130
- Intermediate code, 187–192, 217
  - expression tree, 190–192
  - postfix string, 187–188
  - quadruple, 189–190
  - triple, 188–189
- Interpretation, 7–8
- Interpreter, 3–4, 212–218
  - advantages, 213
  - impure interpreter, 217
  - schematic, 213–214
- Interprocess communication, 331–332, 447–459
- Interprocess message, 447–454
  - in Mach, 458–459
  - in Unix, 456–457
- Interrupt hardware, 291–293
- Interrupt processing, 293
- IO channel, 523
- IO channels, 289–290
- IO device, 526–529
- IO organization, 522–525
- IO processor, *see* IO channel
- IOCS, *see* Physical IOCS, Logical IOCS
- Job class, 352–353
- Kerberos, 647–649
- Knot, 379
- Language migrator, 3
- Language processing, 9
- Language processor, 2–3
  - analysis phase, 9
  - back end, 12
  - definition, 2
  - front end, 12
  - intermediate representation (IR), 12
    - definition, 12
  - pass, 11
  - synthesis phase, 9
- Language translator, 3
- LEX, 31–33
- Lexical analysis, *see* Scanning
- Linker, 221–248, *see also* Object module
  - ENTRY statement, 225
  - external reference, 225
  - EXTRN statement, 225
  - for Intel 8088, 233–245, *see* Intel 8088 linker
  - linking, 225–226, 230–231
  - linking for overlays, 245–248
  - program relocation, 223–225, 229–230
    - definition, 223
    - public definition, 225
- Linking, 225
- Load balancing, 610
- Loader, 221, 248



- absolute loader, 224
- relocating loader, 224
- Locality of reference, 486–487
- Logical address, 481
- Logical device, 531–532
- Logical IOCS, 521, 552–557
- LR parsing, 76, *see also* Bottom up parsing
- Macro
  - attributes, 140
  - conditional expansion, 134, 140–144
  - definition and call, 132–133
  - expansion, 133–137
  - expansion time loops, 134, 141–143
  - expansion time variable, 139–140
  - keyword parameter, 135–137
  - nested macro call, 137–138
  - positional parameter, 134–135
  - semantic expansion, 143–144
- Macro assembler, 158–160
- Macro preprocessor, 145–158
  - data structures, 147–150
  - macro definition processing, 150
  - macro expansion, 153–158
- Mailbox, 454–456
- Memory allocation
  - to a process, 469–471
- Memory compaction, 464
- Memory fragmentation, 302–304, 475–476
  - definition, 302
- Memory handler, 461
- Memory management, 460–518
- Memory protection, 471–473, 492–493
  - bounds registers, 471–472
  - protection keys, 472
- Monitor, 426–436
- MULTICS, 515–517
- Multiprocessors
  - master-slave, 366–367
  - symmetrical multiprocessors (SMP), 368
- Multiprogramming, 287–305
  - hardware support, 288–293
  - program classification, 297–298
  - program priority, 298–302
  - scheduling, 296–302
  - supervisor functions, 295–304
- Networking, 611–615
- Non-preemptible server, 345
- Noncontiguous memory allocation, 479–482
- Object module, 221, 227
- Optimizing transformations, 200–203
- ORIGIN statement, 96, 105, 117
- OS kernel, 315
- OS mechanism, 314
- OS microkernel, 316–317
- OS policy, 314
- Overlay, 245
- Page block table (PBT), 497
- Page map table (PMT), 497
- Page reference string, 500
- Page replacement, 486
- Page replacement policies, 499–505
  - FIFO replacement, 502
  - inclusion property, 504–505
  - LRU replacement, 502–503
  - optimal replacement, 501
- Paging, *see also* Thrashing, Locality of reference, Page replacement
  - address translation, 482–484, 489–490
  - demand paging, 484
  - memory protection, 492–493
  - page, 482
  - page block, 482
  - page fault, 485
  - page traffic, 485
  - sharing of pages, 508–510
- Paging hardware, 488–496
- Paging software, 496–510
- Parbegin–Parend, 333–334
- Parse tree, 22–23, 64–65
- Parsing, 15, 64–84, *see also* Top down parsing, Bottom up parsing
- Password, 592, 593
- Physical address, 481
- Physical IOCS, 521, 530–542
- PL domain, 1
- Prediction, 67
- Preprocessor, 3
- Privileged mode, 290–291
- Problem oriented language, 4
- Procedure call, 194–198

- calling convention, 195–196
- side effect, 194
- Procedure oriented language, 4
- Process, 321–340
  - creation, 322
  - definition, 320
  - dispatching, 354, 358
  - interacting processes, 327–332
    - control synchronization, 329–330
    - data access synchronization, 330
    - definition, 328
    - implementation, 332–340
  - preemption, 354, 358
  - scheduling, 326, 358–360, 362–364
  - state, 322–323
  - termination, 327
- Process control block (PCB), 324–326
- Process precedence, 396
  - implementation, 398
- Process precedence graph, 397
- Process precedence sequence, 397–398
- Process synchronization, 396–443, *see also*
  - Critical section, Semaphores, Critical region, Conditional critical region
  - classical problems, 408–411
  - control synchronization, 396–398
- Producer–Consumer problem, 408–409, 416–417, 421–422
- Producer–Consumer problem, 424, 442
- Program flow graph (PFG), 207
- Program generation, 5–7, 257
- Program instrumentation, 256
- Program interpretation, 9
- Program mix, 297
- Program priority, 298–302
- Program relocation, 473–474
- Program translation, 7, 9
- Programming environment, 262–264
- Programming language
  - data structure, 164
  - data type, 162–163
  - definition, 162
  - execution gap, 2
  - scope rule, 164–165, 168–169
  - semantic gap, 162–165
- Programming language grammar, 19–27
  - alphabet, 19
  - ambiguity, 27
  - classification, 25–26
  - definition, 20
  - handle, 78
  - nonterminal symbol, 20
  - operator grammar, 26
  - operator precedence, 79
  - operator precedence grammar, 79–80
  - production, 20
  - recursive specification, 23–25
  - simple phrase, 78
  - simple precedence, 77
  - simple precedence grammar, 77
  - string, 19
  - terminal symbol, 19
- Protocol, 449
- Race condition, 330, 399
- Readers–Writers problem, 409–410, 417–419
- Readers–Writers problem, 425, 444
- Real time application, 311–313
  - definition, 311
- Real time system, 311–313
- Recovery, 583–584, 638–639
- Recursion, 175–176
- Reduction, 22
- Regular expression, 61–62
- Relocation factor, 224
- Remote procedure call (RPC), 617–619
- Resiliency, 641–642
- Resource knot, 381
- Resource request and allocation graph (RRAG), 373–374
- Response time, 305
- Round robin scheduling, 306
- Scanning, 14–15, 59–63
  - use of automata, 60–63
- Scheduling, 279, 296–297, 306–309, 343–368
  - in multiprocessor OS, 366–368
  - job scheduling, 351–353
  - non-preemptive, 345–348
  - preemptive, 348–350
  - process scheduling, 353–365
    - multi-level scheduling, 362–364
    - multiprogramming, 358–360

- time sharing, 360–362
- Search data structures, 37–52
- Security, 643–649
- Segmentation, 511–517
- Self relocating program, 232
- Semantic analysis, 15–16
- Semantic gap, 1
- Semaphore, 413–419
  - definition, 413
  - implementation, 415–416
- Sentence, 21
- Sentential form, 21
- Shift-reduce parsing, 76, *see also* Bottom
  - up parsing
- Simula, 426–428
- Software tool, 249–269
  - definition, 249
  - program generation, 257
  - program preprocessing, 256
- Source language, 3
- Specification gap, 2
- Spooling, 393
- Stable storage, 581
- Stack, 52–54
- Static binding, 30
- Swapping, 309
- Symbol table, 38
- Syntax analysis, *see* Parsing
- System call, 295
- Table organizations, 41–49, *see also* Hash
  - table
    - binary search, 43–44
    - linked lists, 49
    - sequential search, 42–43
    - tree structured tables, 49–50
- Target language, 3
- Text editor, *see* Editor
- Thrashing, 487–488
- Threads, 336–342
  - Kernel level threads, 338–339
  - User level threads, 339–340
- Throughput, 296
- Time sharing, 305–309
  - memory management, 309
  - scheduling, 306–309
- Time slice, 307
- Time-stamp, 621–622
- Token, 626
- Top down parsing, 65–75
  - LL parsing, 73–74
  - recursive descent parsing, 71–73
  - without backtracking, 69–70
- Turn around time, 279
- Two phase commit, 640–641
- Unix
  - access control list, 595
  - buffer cache, 558
  - device driver, 541–542
  - directory mounting, 585–586
  - file allocation, 584–585
  - file descriptor, 558
  - file processing, 558–559
  - file sharing, 586
  - file system, 584–586
  - i-node, 558
  - interprocess message, 456–457
  - memory management, 511
  - process, 334–335
  - process management, 365–366
- User interface (UI), 264–269
- Virtual memory (VM), 480–482, *see also* Paging
  - address translation, 480–481
- Virtual memory system, 481
- VTOC, 562
- Wait for graph(WFG), 374
- Working set, 506–507
  - definition, 506
- YACC, 31–34



Second Revised Edition



# Systems Programming and Operating Systems

Extensively rewritten to enhance readability and cohesion. Numerous improvements in the presentation of concepts have been effected throughout; errors and ambiguities have been corrected.

## Highlights of this edition :

- ▶ Added a section on resource allocation and user interface functions.
- ▶ The section on threads has been rewritten.
- ▶ Discussion of working set allocator is revised completely—elaborations are added to the example on working set allocator.
- ▶ New discussion added on IO initiation.
- ▶ Sections on hashing function, garbage collection, code optimisation, relocation and linking concepts, deadlock handling algorithms, system states and many others have been elaborated upon and rewritten to enhance understanding.
- ▶ Chapters on Software Tools, Assembler, Interprocess Communication, File Systems and Protection and Security are substantially rewritten.

Visit us at: [www.tatamcgraw.com](http://www.tatamcgraw.com)

The McGraw-Hill Companies

**Mc  
Graw  
Hill**

**Higher Education**

ISBN-13: 978-0-07-416357-7  
ISBN-10: 0-07-46357-7



9 780074 635797

c/005