

SYSTEMS PROGRAMMING

Srimanta Pal

*Associate Professor
Indian Statistical Institute
Kolkata*

OXFORD
UNIVERSITY PRESS

Contents

Preface v

1.	Scope of Systems Programming	1	1.3	Concept of Hardware	2
1.1	Introduction	1	1.4	Machine Structure	3
1.2	Computers and Human Beings	2	1.5	Some System Software Concepts	4
			1.6	Operating Systems	6
			1.7	System Repairing and Debugging Tools	8

PART 1

SYSTEMS PROGRAMMING BACKGROUND

2.	Prerequisites of Systems Programming	11	3.10	Language Translators	43
2.1	Introduction	11	3.11	Operating Systems	46
2.2	Systems	11	3.12	Distributed System	48
2.3	Popular Terms and Concepts	13	3.13	Tools	48
2.4	Number Systems and their Representation	14	3.14	Database System	49
2.5	Components of A Computer Program	20	3.15	Life Cycle of a Source Program	49
2.6	Programming Techniques	21	3.16	Different Views on the Meaning of a Program	50
2.7	Data Structures	23	3.17	System Software Development	50
2.8	Design of Algorithms	29	3.18	Recent Trends in Software Development	50
2.9	Analysis of Algorithms	32	3.19	Levels of System Software	51
3.	Overview of System Software	36	4.	Machine Structures	53
3.1	Introduction	36	4.1	Introduction	53
3.2	Software	36	4.2	John Von Neumann Machine Structure	53
3.3	Software Hierarchy	37	4.3	A General Approach to Machines	57
3.4	Systems Programming	37	4.4	Numbered Register-Based Machine	58
3.5	Embedded Systems Programming	39	4.5	Instructional Computers	62
3.6	Machine Structure	39	4.6	Accumulator-Based Computers	65
3.7	Interfaces	39	4.7	Classification of Computers	67
3.8	Address Space	40	4.8	Microprocessor Evolution	69
3.9	Computer Languages	42	4.9	The Architecture of A Microcomputer	70
			4.10	CISC Architecture-Based Computers	71
			4.11	RISC Architecture-Based Computer	82

5.	System File and Library Structures	86	5.5	Object Code	93
5.1	Introduction	86	5.6	Object Files	98
5.2	Library and File Organization	86	5.7	Standard Object File Structures	103
5.3	Design of a Record	87	5.8	Executable File	115
5.4	Source Program/Data File Structure	92	5.9	Standard Executable File Structures	116
			5.10	Libraries	122
			5.11	Image File Structures	124

PART 2

LOW-LEVEL TRANSLATORS

6.	Machine and Mnemonic Languages	129	8.	Programming in Assembly Language: an Illustrative Approach	195
6.1	Introduction	129	8.1	Introduction	195
6.2	Machine Languages	129	8.2	Features	195
6.3	Mnemonic Languages	130	8.3	The Intel 8086 Instruction Set	196
6.4	Machine Language vs Mnemonic Language	130	8.4	Interrupts	196
6.5	Machine Operations In Machine Languages	131	8.5	Housekeeping of Interrupt Calls	197
6.6	Programming Techniques In Machine Languages	134	8.6	Programming Techniques	198
6.7	Object Code	173	8.7	Assembly Language Programming Styles	201
6.8	Advanced Concepts	173	8.8	Language Constructs	202
7.	Assembly Languages	175	8.9	How to Run Assembly Programs	203
7.1	Introduction	175	8.10	Standard Assemblers	203
7.2	Roles of Assembly Language	175	8.11	Microsoft Assembler (MASM)	203
7.3	Basis of Assembly Languages	175	8.12	Borland Turbo Assembler (TASM)	205
7.4	Disadvantages	176	8.13	Microasm	207
7.5	Why Learn Assembly Languages?	176	8.14	Netwide Assembler (NASM)	208
7.6	Comparisons	176	8.15	DOS Batch Files for TASM and MASM	208
7.7	Assembly Language Applications	177	8.16	Memory Models	209
7.8	Multi-Platform Assembly Programming	178	8.17	Program Illustrations	209
7.9	Assembly Language Formats	178	9.	Assemblers	237
7.10	Assembly Statements	179	9.1	Introduction	237
7.11	Basic Operations	180	9.2	Basic Tasks	237
7.12	Importance of Registers	184	9.3	Classification of Assemblers	238
7.13	Macros	184	9.4	Working Principles	238
7.14	Structured Assembly Language	185	9.5	Problems in Manual Assembly	241
7.15	Elements of Assembly Programs	185	9.6	General Problems in Assembler	241
7.16	Programming Techniques	185	9.7	General Procedure for Assembler Design	241
7.17	Assembly Language Subroutines	192	9.8	A Simple Assembly Process	242
			9.9	Assembly Process	245
			9.10	Design Issues	253
			9.11	Assembler Features	258

9.12	Assembler Design Criteria	259	12.9	Generalized one-Pass Macro Processors	361
9.13	Types of Assemblers	259	12.10	Applications of Macro Processors	367
9.14	Two-Pass Assemblers	259	12.11	Macro Command Interpreter	367
9.16	One-Pass Assemblers	280			
9.17	Multi-Pass Assemblers	286	13.	Linkers	372
9.18	Advanced Assembly Process	286	13.1	Introduction	372
9.19	Variants of Assemblers	288	13.2	Principles of Linking	373
10.	Macros and Macro Languages	293	13.3	A Simple Linking Process	374
10.1	Introduction	293	13.4	Basic Linking Tasks and Procedure	375
10.2	Definitions	295	13.5	Memory Management During Linking	376
10.3	Advantages of Using a Macro	295	13.6	Symbols and Symbol Resolution	378
10.4	Macro vs Procedure	295	13.7	Binding	379
10.5	Development of Macro Languages	296	13.8	Resolving External References	380
10.6	Local Variables in a Macro	297	13.9	Relocation and Code Modification	380
10.7	Macros Without Parameters	297	13.10	Linking Methods	385
10.8	Concept of Parameters in Macros	300	13.11	Static Linking	385
10.9	Formal vs Actual Parameters in Macros	302	13.12	Dynamic Linking	385
10.10	Concatenation of Macro Parameters	302	13.13	Library Linking	389
10.11	Parameter-Passing Mechanism in Macros	302	13.14	Object Linking and Embedding (OLE)	392
10.12	Conditions in Macro Definitions	307	13.15	Module Linking	392
10.13	Macro Calls Within Macro Definitions	311	13.16	Design of Linkers	393
10.14	Macro Instructions Defining Macros	313	13.17	Linker Command Languages	398
10.15	Case Studies	314	13.18	Automatic Library Searching	399
11.	Macro Programming: An Illustrative Approach	322	13.19	Some Popular Linkers	400
11.1	Introduction	322	14.	Loaders	405
11.2	Features	322	14.1	Introduction	405
11.3	Macro Programming Techniques	322	14.2	Programs in Memory	405
11.4	Program Illustrations	325	14.3	Principles of Loading Operation	405
12.	Macro Processors	335	14.4	Linkers vs Loaders	407
12.1	Introduction	335	14.5	Loading Shared Libraries From Applications	408
12.2	Functions of a Macro Processor	335	14.6	Different Loading Schemes	408
12.3	Basic Tasks of a Macro Processor	336	14.7	Sequential and Direct Loaders	409
12.4	Design Issues of Macro Processors	336	14.8	Compile-and-Go Loaders	409
12.5	Features	336	14.9	General Loader Schemes	410
12.6	Macro Processor Design Options	337	14.10	Absolute Loaders	411
12.7	Two-Pass Macro Processors	339	14.11	Relocating Loaders	415
12.8	One-Pass Macro Processors	350	14.12	Practical Relocating Loaders	422
			14.13	Linking Loaders	423
			14.14	Relocating Linking Loaders	427
			14.15	Overlay	429
			14.16	Binder	433
			14.17	Dynamic Loader	434
			14.18	Automatic Library Search	434
			14.19	Complexity	435
			14.20	Some Popular Loaders	435
			14.21	Graphics Loaders	441

15. Object Code Translators	444	15.3 Object Code Translators	447
15.1 Introduction	444	15.4 Translation Process	447
15.2 Binary Code Translators	444	15.5 Hybrid Method	449
		15.6 Applications	450

PART 3

HIGH-LEVEL TRANSLATORS

16. Overview of High-Level Translators	453	17.19 Object-Oriented Programming Languages	491
16.1 Introduction	453	17.20 Functional Programming Languages	493
16.2 Programming Languages	454	18. Design of Compilers	495
16.3 Compilers	455	18.1 Introduction	495
16.4 Variant of Compilers	456	18.2 Compilation Process	496
17. High-Level Programming Languages for Compilers	458	18.3 Other Programs with Compilers	497
17.1 Introduction	458	18.4 Translation of a Statement	498
17.2 Nature	458	18.5 Compiler Construction Tools	498
17.3 Needs	459	18.6 Lexical Analysis	499
17.4 Structured Language	459	18.7 Syntax Analysis	505
17.5 Features	460	18.8 Basic Parsing Techniques	505
17.6 Regular Expressions	460	18.9 Semantic Analysis	523
17.7 Finite Automata	461	18.10 Parameter Passing	526
17.8 Grammars	462	18.11 Intermediate Code Generation	528
17.9 Context-Free Grammars	463	18.12 Code Generation	531
17.10 Ambiguous and Unambiguous Grammars	470	18.13 Code Optimization	532
17.11 Left-Recursive Grammar	471	18.14 Optimizing Compilers	533
17.12 Grammar Classes	471	18.15 Some Popular Compilers	534
17.13 BNF to Parsing Table	472	19. Compiler-Compilers	538
17.14 Some Terminology Used in Parsing	481	19.1 Introduction	538
17.15 Parsing Table Construction	481	19.2 Design of Compiler Generators	539
17.16 Predictive Parsing Table Construction	482	19.3 Popular Compiler Generators	542
17.17 LR Parsing Table Construction	483	19.4 Lex	545
17.18 Operator Precedence Grammars	485	19.5 Yacc	549
		19.6 Inside of Lex and Yacc	552
		19.7 Compiler Design Toolbox	553
		19.8 Universal Compiler-Compilers	554

PART 4

LOWEST-LEVEL TRANSLATORS

20. Principles of Operating Systems	567	20.5 Interrupts	588
20.1 Introduction	567	20.6 Structural Overview of Operating Systems	594
20.2 The Scope of Operating Systems	567	20.7 State Transition of Concurrent Processes	596
20.3 The Functions of Operating Systems	569		
20.4 Memory Management Schemes	571		

20.8	Higher-Level Synchronizing Tools	597	20.12	Device Management	616
20.9	Deadlocks	599	20.13	Buffer Allocation Algorithm	628
20.10	Handling Deadlocks	601	20.14	Case Studies	629
20.11	Scheduling	610			

PART 5

SYSTEM TOOLS

21.	Editors	649	22.6	Debugging Using Hardware	672
21.1	Introduction	649	22.7	Use of Debugger in Unix C	672
21.2	Document Editor	649	23.	System Administration	675
21.3	The Design of a Text Editor	651	23.1	Introduction	675
21.4	Sound Editor	661	23.2	System Administrators	675
21.5	Case Studies	662	23.3	System Installation	677
22.	Debuggers	667	23.4	System Configuration	678
22.1	Introduction	667	23.5	On-Going Process	678s
22.2	Types of Errors	667	23.6	System Administration for DOS	679
22.3	Debugging Procedures	668	23.7	System Administration for Windows	679
22.4	Classification of Debuggers	669	23.8	System Administration for UNIX	681
22.5	Dynamic/Interactive Debugger	669	Appendix	683	
			Bibliography	708	
			Index	713	