CHAPTER 4: SYSTEM SOFTWARE

4.1 PURPOSES OF AN OPERATING SYSTEM

4.1.1 Resource Optimisation

How OS Maximises Resources:

- CPU scheduling
- Memory management
- I/O optimisation
- File system management

4.1.2 User Interface

Purpose: Hides complexities of hardware from user. Allows users to interact with application programs without knowing hardware details.

Types:

- Command-line interface (CLI)
- Graphical User Interface (GUI)
- Touch interface

4.1.3 Process Management

Process: A program being executed which has an associated Process Control Block (PCB) in memory.

Process States:

- **Ready:** New process arrived, PCB created
- **Running:** Has CPU access
- **Blocked:** Cannot progress until event occurs

PCB Contents:

- Process state
- Program counter
- CPU registers
- Memory management information
- I/O status information

4.1.4 Scheduling Routines

First-Come-First-Served (FCFS):

- Non-preemptive
- FIFO queue
- Simple but may cause long waits

Round Robin:

- Preemptive
- Allocates time slice to each process
- No prioritization
- Fair but may waste CPU on context switching

Priority-Based:

- Most complex
- Priorities re-evaluated on queue change
- Can be preemptive or non-preemptive

Shortest Job First:

- Non-preemptive
- Processes with shortest execution time first
- Optimal for minimizing average waiting time

Shortest Remaining Time:

- Preemptive version of Shortest Job First

4.1.5 Memory Management

Paging:

- Process split into pages
- Memory split into frames
- All pages can be loaded into memory

Virtual Memory:

- No need for all pages in memory
- CPU address space larger than physical
- Address resolved by Memory Management Unit (MMU)

Benefits:

- Large programs can run with less physical memory
- More programs can run simultaneously

Disk Thrashing:

- Perpetual loading/unloading of pages
- Occurs when too many processes compete for memory
- Performance severely degraded

4.1.6 OS Structure

User Mode:

- Available for users and applications
- Limited access to system resources

Kernel/Privileged Mode:

- Sole access to parts of memory
- Access to certain system functions
- Only OS components run in this mode

4.2 TRANSLATION SOFTWARE

4.2.1 Compilation Stages

Lexical Analysis:

- Converts sequence of characters to sequence of tokens
- Identifies keywords, identifiers, operators

Syntax Analysis:

- Checks code for grammar mistakes
- Identifies syntax errors

Code Generation:

- Generates intermediate code after syntax analysis

Optimization:

- Improves efficiency of code
- Removes redundant operations

4.2.2 Interpreters vs Compilers

Feature	Compiler	Interpreter
Translation	Entire program before execution	Line-by-line
Output	.exe file	No executable
Execution speed	Faster	Slower
Error reporting	All errors at end	Stops at first error
Debugging	Difficult	Easier

4.2.3 BNF (Backus-Naur Form)

Purpose: Formal mathematical way of defining syntax unambiguously.

Components:

- Set of terminal symbols
- Set of non-terminal symbols
- Set of production rules

Example:

<TEXT>

<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<integer> ::= <digit> | <integer> <digit>

4.2.4 Reverse Polish Notation (RPN)

Definition: A method of representing expressions without brackets or special punctuation. Uses postfix notation where operator is placed after variables.

Example:

- Infix: A + B
- RPN: A B +

Advantages:

- No need for brackets
- No need for operator precedence
- Simpler for machine to evaluate
- No backtracking needed

Evaluation:

- Use stack
- When operator encountered, pop two operands, apply operator, push result

---

Revision #1
Created 2026-03-16 12:16:18 UTC by Samuel Lee
Updated 2026-03-16 12:16:33 UTC by Samuel Lee