

History of Operating Systems

- Reading: Dijkstra, "Cooperating Sequential Processes" (1965)
- Question: What is an OS?
- Question: Why bother? Everything done with OS can be done without.

1

OS Generation 0

Computer usage:

- Queue or sign-up sheet for job scheduling
- Hope computer is working during your time slot
- Load, run, debug programs while on machine
- Debugging via front panel:
 - PC — Program Counter
 - MAR — Memory Address Register
 - A — Accumulator
 - PSW — Processor Status Word

2

OS Generation 1: Early Batch

- Operator assisted
 - Distance programmers from system
 - Streamline ordinary computer usage
- Jobs submitted to operator
 - Clustered in batches
 - Card to tape (spooling by hand)
 - Output to tape (then to card punch, line printer by operator)

3

Fortran Monitor System (1957; IBM 709)

one of the first batch OSS

- Monitor (job launch, termination, resource allocation, job command language)
- Assembler
- FORTRAN
- Link-loader
- Subroutine library

4

Data Channels...

Early use of multiple processors for DMA

- Once initiated, independent of CPU
- Buffered I/O

5

... Data Channels

ist overlap of I/O and computation

- Accessible only via assembler
- FORTRAN, COBOL, ALGOL had no language constructs for parallel operations, synchronization
- In assembler
- I/O call
- computation
- wait loop

6

Interrupts

- Signal lines to CPU
- Checked (in priority order) once per execution cycle
- Each interrupt tied to a specific low-core routine
- Eliminates wait loops
- Monitor executes a global wait loop, instead of local wait loops in applications
- Big efficiency gain

7

Interrupt Routines

- Must be able to suppress interrupts in critical code (E.g., laying down interrupt context, cleaning up)
- Must be re-entrant—able to process multiple interrupts simultaneously (multi-threaded)
- Pure code with data stack for context

8

Accounting and Billing

- Added later
- CPU usage; I/O usage

Job deck:

Job Control Language...

```

$JOB user-id
$DATE
$REQ PRINT CLG
$FORTRAN
FORTRAN source
$DATA
data

```

9

... Job Control Language

Not thought of as a *command language*

So, missing standard features of languages: variables, iteration, conditionals, macros,...

(until time-sharing)

Nevertheless, reached peak of complexity: IBM OS/360 JCL

10

11

```

SUMMARY
EXEC      JOB      REGION=(100K,50K)
          PGM=SDUMMAR
          DSNNAME=RIP, 6501, DISP=OLD,
          UNIT=2314, SPACE=(TRK, (1,1,1)),
          VOLUME=SSR=577692
          DSNNAME=SUM, 6501, DISP=(,KEEP),
          UNIT=2314, SPACE=(TRK, (1,1,1)),
          VOLUME=SSR=577692
          DD      SYSABEND
          DD      SYSOUT=A
summary <jan.report >jan.summary

```

IBM OS/360 JCL

Protection

Users are admonished not to

- R/W OS core
- R/W system tape drive
- Read past EOF on batch input tape
- Last two enforced in system I/O routines (but no obligation to use them)

Resident Supervisor

- Interrupt processing
 - I/O support
 - Monitor kernel
- Aggravating space problem for users

Software Protection

O routines check (e.g.)

- operation versus ID
- refuses to read past EOF on input tape

onitor checks against resource limits (e.g., time, I/O) when
ontrol returns

- Operator "protects" against infinite loops

12

13

14

Hardware Protection (c. 1963)

Timer interrupts: protection from infinite loops

Privileged versus user mode

Privileged mode requires privilege bit set in PSW

Required for:

- I/O (only system routines can execute I/O)
- Changing PSW (low core can change priv. bit)

15

16

user code can flip priv. bit only by calling low-core system code
bounds registers: memory access beyond bounds is a privileged
operation—else Interrupt

... OS Generation 2: Multiprogramming
process scheduling

- Use of time slices to provide "fairness"
- Round-robin READY queue

19

OS Generation 2: Multiprogramming...
Interrupts allow control to pass between jobs when current job waits
("blocks") on I/O
Completion interrupt puts original job into READY state

17

Disk File Systems

"Random access" versus sequential access of tapes
Two main problems in multiprogrammed environment:

1. File protection: RWX
2. Synchronization

20

... OS Generation 2: Multiprogramming...

Huge improvement in turn-around time

- CPU not idle during I/O (improved throughput)
- Short jobs can complete during long-job execution (constant throughput; smaller average turn-around time)

18

File Protection: RWX

- No protection (e.g., DOS, Windows)
- Group-based protection (e.g., Unix, HP)
- Access-Control Lists (ACL)
 - Inherit access through file hierarchy (e.g., Windows NT)
 - independent per object

21

Synchronization

Exclusive/Shared access locks

- Temporary (per transaction)
- Long-term (per open)

22

Other synchronization problems

In multiprogramming environment

- Race conditions: Who gets resource first?
- Deadlock: Everyone gets only part of what they need
- Solution: semaphores
At bottom, supported by hardware (usually test-and-set instruction)

23

Automated Spooling to/from Disk

- Speedup of central system
- Improved job scheduling

24

OS Generation 3: Time Sharing

Project Whirlwind, MIT

Goal: Universal flight trainer/simulator in a real-time computer system

This goal is yet to be fully achieved (VR)

Project cost: \$8M (16 × ENIAC)

25

Whirlwind's Main Spinoffs

- SAGE air-defence system (which led to SABRE airline reservation system, 1963)
- Core memory (1953)
- Transaction processing, on-line computing industry (POS, ATMs)
- Time sharing
- Digital Equipment Corporation
- Workstations

26

Core Memory

Mercury delay lines were too slow

Whirlwind needed to execute 100,000 instructions per second

CRTs (Williams tubes) not sufficiently reliable

Jay Forrester: use new magnetic ceramic (prototyped, end 1951; production ready, 1953)

MIT received patent and royalties, about \$500,000

Core replaced all other memory within 5 years

27

Whitwind's Other Firsts

- Use of typewriter terminals
- CRTs for display of text and graphics

28

SAGE Project . . .

When money dried up for Whitwind, Forrester and IBM pitched system to DOD for air defence

- IBM built 23 Whitwind computers (IBM AN/FSQ-7)
- 49,000 vacuum tubes each
- each managing real-time oversight of a sector via radar
- each duplexed for reliability

29

. . . SAGE Project

- cost: \$8B
- obsolete on installation (1963), ICBMs
- Led directly to SABRE

30

SABRE . . .

airline reservation/management system, joint project IBM and American Airlines
before SABRE all airline reservations done by hand
enabled expansion of airlines

implemented 1960–1963

31

. . . SABRE

- IBM Selectric terminals (first ball-type typewriters, "golf-balls")
- Hot backup dual (from SAGE)
- Random-access disk for transactions
- Handled 10M transactions per year
- Automated reservations universal by 1970

32

Current Transaction Processing Systems

- Airline reservations, hotel, car rental
- Banking, ATMs
- EFT
- Just-in-time manufacturing/inventory
- Electronic commerce
 - 1,000 TPS systems
 - Greater impact on economy than PCs—new way of doing business

33

Observation

new technology grows on top of old
time bank using TP for ATM support may still use batch processing for
roll, inventory, billing, account statements, cheques...
Automated 19th-century practices

34

Time Shared OS

First proposed (and patented) by Chris Strachey, 1959
"Programmer's Workbench" with card reader, printer connected to shared central system

Reading: Licklider's 1960 "Man-computer symbiosis"

36

MIT Whirlwind

Contributed:

- Concept of real-time processing
- Typewriter terminals; graphics display
- Human-computer interaction

37

Third-Generation Architecture

- contributed
- Interrupts
- Time slices
- Disk drives
- Multi-programming (e.g., protection, synchronization)

38

Dartmouth Time-Sharing System (1964)

- John Kemeny, Thomas Kurtz
- Goal: provide computer power for educating ordinary students (not CS students)
- Constraint: simplicity in usage
- Implemented within the year 1964 on GE 235

39

BASIC

- Beginner's All-purpose Symbolic Instruction Code
- Implemented by a small team of undergraduates within a few months (compare FORTRAN)
- Aimed at time sharing, hence interpreted

40

Popularity of BASIC

ce to

- Popularity of time-shared mainframes (later minicomputers, PCs)
 - "Demographics" of computer usage
- Far more beginners than anyone else from 1964 till present

41

MIT Project MAC

- At MIT time-sharing advocated by John McCarthy
- MAC started with \$3M from ARPA of DOD Advanced Research Projects Agency

42

Compatible Time-Sharing System—CTSS

- student/teaching system
- IBM 709
- 30 simultaneous terminal users
- "compatible" with FMS

43

Compatible Time-Sharing System—CTSS

CTSS demonstrated utility of time-sharing for software development

- on-line editing, debugging
- typewriter terminals

CTSS did time-sharing *without* multiprogramming

- Only one user process active at a time in CPU+memory
- No need for synchronization...

44

CTSS Memory Management

- User programs swapped from drum one at a time into "B core" of 32K words
- "Onion-skin" algorithm:
If old program's space not re-written, it would not be re-read on restart
- Nevertheless, heavy penalty for context switch:
Max. 250ms swap time for all 32K words

45

CTSS Processor Management

Ready → Running

- Queues 0...7
- Time quantum for queue k is $2^k \times 500$ ms (Max. continuous run time)
Poor response time!
- Scheduling favored interactive processes:
If command incomplete at quantum, put on queue $k+1$; else on queue $k-1$
- Lower queues get priority scheduling

46

CTSS Processor Management

state: no wait state for file I/O!

- I/O buffers in B core cannot be released
- File I/O fast compared with context switch

47

MULTICS (Multiplexed Information and Computing Service)

- MIT, GE, Bell Labs
- Initiated 1964, operational 1969
- Goal: realize the "computer utility"
 - Remote usage of shared central computer system via data communications network
 - Analogy to power grid of electric utility
 - One origin of computer networks...

48

Grosch's Law

-

49