Origins of Operating Systems
OS/360

Martin Grund
HPI
Table of Contents

• IBM System 360
• Functional Structure of OS/360
• Virtual Machine Time Sharing
Welcome to Big Blue
IBM System 360

- In 1964 IBM announced the IBM-360 family for computers
  - All machines, despite their differences, had the same user instruction set
  - Different operating systems available for these machines
  - Only midrange and high-end system run OS/360
- IBM introduced the new term of hardware architecture
- In 1970 IBM announced System 370 with hardware virtual memory support
IBM System 360

- High-end machines established 32 bit as standard for computers
- Virtual Memory Support – hardware support for dynamic address translation
- Within ten years S/360 achieved standard status
- Flashback prices:
  - 1970 – $279/MB hard disk
  - 1980 - $35/MB hard disk | $50.000 /MB DRAM
IBM System 360 Specials

• Introduced 8bit entities
• Introduction of 32 or 64 bit floating point words based on a hexadecimal base
• Variable length strings using length field in the first byte
• All registers are universal registers – accumulators as well as address registers
• Registers use 32 bit, 24 bit for addressing -> 16MB
IBM S/360 - Pictures
IBM OS/360 Introduction

- One OS for different application areas
- Upstream compatible
- 1.000.000 program instructions, one of the most complex software ever written
- $ 50.000.000 development costs -> double the price of the Manhattan Project
- Different OS for different hardware -> small, mid-range, high-end
- MFT – small systems | MVT – high-end systems
- Introduction of JCL
OS/360 – Functional Structure

• OS/360 is one of the first “second generation” operating systems

• Primary Objective: accommodate an environment for diverse applications and operating modes

• Secondary objectives:
  • Throughput, Response Time, Productivity, Adaptability, Expandability: Support for many languages, configurations, operating modes
OS/360 – Design Concepts

- Every Program is either executed in Supervisor State or Problem State
  - S-State: for control program to call different other modules
  - P-State: all programs operate in
- Task – combination of program and data

![Diagram of tasks and program states]
OS/360 – Design Concepts

- Multiprogramming: for one CPU several tasks are defined
- Multiprocessing: one task utilizes several CPUs
- Multitasking allows both manners of operating
- Common System Elements:
  - Translators for programming languages
  - Linkage Editor to combine individually translated programs into one executable
  - Service Programs
  - Control Program
OS/360 – Control Program

- Main important part of the OS, contains
  - Master scheduler – control communication between operator and system
  - Job scheduler – formalizes job into task, allocation of devices, initial execution for each job step, write job output
  - Supervisor – memory management, module loading, control execution, logging, monitoring
OS/360 – System Generation

- To support as many operating modes as possible, almost everything can be configured.
- Each configuration leads to a new system generation.
- System generation covers:
  - Support for installed libraries.
  - Defining multi-task/single-task environments.
  - Defining level of concurrent jobs.
- Stage I – parse generation instructions into job stream.
- Stage II – load assembler, linkage editor, catalog library.
Job & Task Management

- Each message arriving to the system is an independent task -> many messages can arrive at the same time
- Control program must decide which task to execute on the status of other tasks
- Basic unit of work is a JOB – can be divided into job steps -> each job step is a task
- All tasks to be executed are treated independent
Program Structure

- All Jobs are defined using the JCL – Job Control Language
- First statement of a job is an identifier for naming, priority definition and other conditions to be matched
- Job Step Statement defines the execute part and a data definition for I/O
- Four program structures to be defined:
Program Structure

• Simple Structure
  • One load module denotes to the entire program

• Planned Overlay Structure
  • Main Program is a single load module, but not all functionality is needed and not loaded into main storage

• Dynamic Overlay Structure
  • Load modules are called dynamically. Control is handed over using LINK, XCTRL and LOAD

• Dynamic Parallel Structure
  • Load modules are executed in parallel (modules declared as note reusable, serially reusable and reenterable)
Program Structure

Simple Structure

Planned Overlay using LINK/RETURN

Dynamic Overlay using LINK / XCTL
JCL – Job Control Language

- Scripting language to instruct the Job Entry Subsystem on how to run a batch program or start a subsystem

- JCL is defined using:
  - One or more 80-byte records
  - Must be uppercase

- It is possible to plan whole software execution steps on the forehead, independent of physical environment (using DD)

- Main instructions: JOB, EXEC, DD, PROC
JCL – Job Control Language

- //jobname JOB (accounting information), CLASS=x, MSGCLASS=x, REGION=nK, TIME=(m,s)
  - CLASS – priority definition (environmental settings)
  - MSGCLASS – how the output is printed or filed
  - REGION – size of virtual memory to use
  - TIME – total CPU usage allowed by the job, max 1439 minutes or 1440 (untimed)

- //stepname EXEC
  PGM=progname, PARM='parm', COND=condition, REGION=n, TIME=(m,s)
  - PGM – name of the module to be executed
  - PARM – optional parameter to the module
  - COND – defines condition when the job is NOT run
  - REGION and TIME as described above
JCL - Example

//PASIVP JOB CLASS=A,MSGLEVEL=(1,1),REGION=256K
//RUN1 EXEC PASCAL,PARM.COMPILE='COMPILATION OPTION LIST'
//COMPILE.INPUT DD *
PROGRAM Hello (Output) ;
BEGIN
  WRITELN(' Hello World!');
END.

//RUN2 EXEC PASCAL,PARM.COMPILE='COMPILATION OPTION LIST'
//COMPILE.INPUT DD *
PROGRAM fib_demo(OUTPUT) ;
TYPE pos_int = 0..30 ;
VAR i : pos_int ;
  time : INTEGER ;
FUNCTION fibonacci(j :pos_int) : INTEGER ;
(*To evaluate fibonacci # j, for j >= 0,
  subject to integer overflow*)
BEGIN
OS/360 – Job Management

- Allocation of I/O devices
- Analysis of the Job Stream
- Overall Scheduler
- Direction of setup activities
- Job Management turns each task to task management – using job scheduler and master scheduler
- Scheduling done either sequential or priority based
- Either Single Job Instantiation or Multi Job Instantiation
Task Management

- Single Task Management
  - Job scheduler operates as a task and executes loaded modules as subtasks (using LINK or XCTL)
  - In the end supervisor sends callback to job scheduler

- Multi Task Management
  - Main focus – resources management
  - Introduces CPU Manager – task dispatcher, task queue
  - Queue either ordered by sequence or by priority
  - All elements of the queue either in READY or WAIT state
Task Management

- **WAIT** – a task waits for an event to occur
  - Events are submitted using the POST macro
  - e.g. Read from I/O makes the task wait until ready
OS/360 Resume

- Delivery too late
- Exceeded Budget
- Needed more hardware resources than planned
- Was replaced within a few years by other OS from IBM
- But:
  - Definition of new ways for Computer Science
    - Problem-State and Supervisor-State -> Kernel-Mode and User-Mode
    - LINK or XCTRL origin for today’s FORK command
  - Creation of a general architecture family
OS/360 Lessons Learned

- Many concepts for today's operating systems were introduced in OS/360
- Complexity of installing/running an OS were reduced heavily
- Today OS/360 can be run on almost any machine using the Hercules emulator and is freely available as public domain software
Virtual Machine Time Sharing

- Approach to maximize the utilization of a single system
- Simulate the whole underlaying system
  - In the beginning: S/360 40 with new modification for memory address translation mechanisms (later S/360 67)
- Combination of VMCP/CMS on S/360 40 was developed on the MIT
- Virtual Machine Control Program – provide functionality for time-sharing, resource allocation, system handling
- CMS – Simple single user Monitoring System for S/360
VM Environment

- VM simulates the whole system
- Support for many OS like OS/360, DOS/360, RAX...
- 24bit addressing region used to assign virtual memory to each guest system
- I/O partitioning – VM devices are mapped onto real devices
- Requirements: not real-time dependent, almost same machine instruction set used by OS, guest OS must support privileged and unprivileged mode
- VM must be isolated and CP must be protected
VM – Paging Explained

- Since cumulated size of main storage for all VMs exceeds main storage of the host, virtual addressing is needed
- Virtual main storage is divided into 4096 byte blocks (pages) -> all active pages are kept in main memory, other pages are swapped to secondary memory
- Dynamic address translation translates address into the current real memory
- Each VM has its own main memory
Demo

- Start OS/360 – execute simple Job from the main terminal to compile and run a Pascal program
- Show S/360 with TSO to show VM mechanisms
Q & A

Any Questions?
Literature

[4] System/360 Announcement -
   http://www-03.ibm.com/ibm/history/exhibits/mainframe/mainframe_PR360.html
[5] Computer History IBM 360/370/390 -
   http://www.beagle-ears.com/lars/engineer/comphist/ibm360.htm
[10] IBM Systems Journal – A virtual machine time sharing System