Operating System Concepts

Agenda

- OS Types
- Terminologies
  - Multi-Programming
  - Multi-Tasking
  - Multi-Threading
  - Multi-Processing
  - Multi-User
- Process Life Cycle
- CPU Scheduling
- Scheduling Algorithms
- Booting

Classification of OS

- OS can be categorized based on the target system (computers).
  - Mainframe systems
  - Desktop systems
  - Multi-processor (Parallel) systems
  - Distributed systems
  - Hand-held systems
  - Real-time systems

Mainframe systems

Resident Monitor

- Early (oldest) OS resides in memory and monitor execution of the programs. If it fails, error is reported.

- OS provides hardware interfacing that can be reused by all the programs.

Batch Systems

- The batch/group of similar programs is loaded in the computer, from which OS loads one program in the memory and execute it. The programs are executed one after another.
- In this case, if any process is performing IO, CPU will wait for that process and hence not utilized efficiently.

Multi-Programming

- In multi-programming systems, multiple program can be loaded in the memory.
- The number of program that can be loaded in the memory at the same time, is called as "degree of multi-programming".
- In these systems, if one of the process is performing IO, CPU can continue execution of another program. This will increase CPU utilization.
- Each process will spend some time for CPU computation (CPU burst) and some time for IO (IO burst).
    - If CPU burst > IO burst, then process is called as "CPU bound".
    - If IO burst > CPU burst, then process is called as "IO bound".
- To efficiently utilize CPU, a good mix of CPU bound and IO bound processes should be loaded into memory. This task is performed by an unit of OS called as "Job scheduler" OR "Long term scheduler".
- If multiple programs are loaded into the RAM by job scheduler, then one of process need to be executed (dispatched) on the CPU. This selection is done by another unit of OS called as "CPU scheduler" OR "Short term scheduler".

Multi-tasking OR time-sharing

- CPU time is shared amoung multiple processes in the main memory is called as "multi-tasking".
- In such system, a small amount of CPU time is given to each process repeatedly, so that response time for any process < 1 sec.
- With this mechanism, multiple tasks (ready for execution) can execute concurrently.
- There are two types of multi-tasking:
    - Process based multitasking: Multiple independent processes are executing concurrently. Processes running on multiple processors called as "multi-processing".
    - Thread based multi-tasking OR multi-threading: Multiple parts/functions in a process are executing concurrently.

Multi-user

- Multiple users can execute multiple tasks concurrently on the same systems. e.g. IBM 360, UNIX, Windows Servers, etc.
- Each user can access system via different terminal.
- There are many UNIX commands to track users and terminals.
    - tty, who, who am i, whoami, w

Desktop systems

- Personal computers -- desktop and laptops
- User convinence and Responsiveness
- Examples: Windows, Mac, Linux, few UNIX, ...

Multiprocessor systems

- The systems in which multiple processors are connected in a close circuit is called as "multiprocessor computer".
- The programs/OS take advantage of multiple processors in the computer are called as "Multiprocssing" programs/OS.
    - Windows Vista: First Windows OS designed for multi-processing.
    - Linux 2.5+: Linux started supporting multi-processing.
- Modern PC architectures are multi-core arch i.e. multiple CPUs on single chip.
- Since multiple tasks can be exeuted on these processors simultaneously, such systems are also called as "parallel systems".
- Parallel systems have more throughput (Number of tasks done in unit time).
- There are two types of multiprocessor systems:
    - Asymmetric Multi-processing
    - Symmetric Multi-processing

Asymmetric Multi-processing

- OS treats one of the processor as master processor and schedule task for it. The task is in turn divided into smaller tasks and get them done from other processors.

Symmetric Multi-processing

- OS considers all processors at same level and schedule tasks on each processor individually.
- All modern desktop systems are SMP.

Distributed systems

- Multiple computers connected together in a close network is called as "distributed system".
- Its advantages are high availability (24x7), high scalability (many clients, huge data), fault tolerance (any computer may fail).
- The requests are redirected to the computer having less load using "load balancing" techniques.
- The set of computers connected together for a certain task is called as "cluster". Examples: Linux.

Handheld systems

- OS installed on handheld devices like mobiles, PDAs, iPODs, etc.
- Challenges:
  - Small screen size
  - Low end processors
  - Less RAM size
  - Battery powered
- Examples: Symbian, iOS, Linux, PalmOS, WindowsCE, etc.

Realtime systems

- The OS in which accuracy of results depends on accuracy of the computation as well as time duration in which results are produced, is called as "RTOS".
- If results are not produced within certain time (deadline), catastrophic effects may occur.
- These OS ensure that tasks will be completed in a definite time duration.
- Time from the arrival of interrupt till begin handling of the interrupt is called as "Interrupt Latency".
- RTOS have very small and fixed interrupt latencies.
- RTOS Examples: uC-OS, VxWorks, pSOS, RTLinux, FreeRTOS, etc.

Process Life Cycle

Process States

- New
  - New process PCB is created and added into job queue. PCB is initialized and process get ready for execution.

- Ready
  - The ready process is added into the ready queue. Scheduler pick a process for scheduling from ready queue and dispatch it on CPU.
- Running
  - The process runs on CPU. If process keeps running on CPU, the timer interrupt is used to forcibly put it into ready state and allocate CPU time to other process.
- Waiting
  - If running process request for IO device, the process waits for completion of the IO. The waiting state is also called as sleeping or blocked state.
- Terminated
  - If running process exits, it is terminated.
- Linux: TASK_READY/TASK_RUNNING (R), TASK_INTERRUPTIBLE (S), TASK_UNINTERRUPTIBLE (D), TASK_STOPPED(T), TASK_ZOMBIE (Z), TASK_DEAD (X)

Types of Scheduling

Non-preemptive

- The current process gives up CPU voluntarily (for IO, terminate or yield).
- Then CPU scheduler picks next process for the execution.
- If each process yields CPU so that other process can get CPU for the execution, it is referred as "Co-operative scheduling".

Preemptive

- The current process may give up CPU voluntarily or paused forcibly (for high priority process or upon completion of its time quantum)

Scheduling criterias

CPU utilization: Ideal - max

- On server systems, CPU utilization should be more than 90%.

- On desktop systems, CPU utilization should around 70%.

Throughput: Ideal - max

- The amount of work done in unit time.

Waiting time: Ideal - min

- Time spent by the process in the ready queue to get scheduled on the CPU.
- If waiting time is more (not getting CPU time for execution) -- Starvation.

Turn-around time: Ideal - CPU burst + IO burst

- Time from arrival of the process till completion of the process.
- CPU burst + IO burst + (CPU) Waiting time + IO Waiting time

Response time: Ideal - min

- Time from arrival of process (in ready queue) till allocated CPU for first time.

Scheduling Algorithms

FCFS

- Process added first in ready queue should be scheduled first.
- Non-preemptive scheduling
- Scheduler is invoked when process is terminated, blocked or gives up CPU is ready for execution.
- Convoy Effect: Larger processes slow down execution of other processes.

SJF

- Process with lowest burst time is scheduled first.
- Non-preemptive scheduling
- Minimum waiting time

SRTF - Shortest Remaining Time First

- Similar to SJF - but Preemptive scheduling
- Minimum waiting time

Priority

- Each process is associated with some priority level. Usually lower the number, higher is the priority.
- Preemptive scheduling or Non Preemptive scheduling
- Starvation
    - Problem may arise in priority scheduling.
    - Process not getting CPU time due to other high priority processes.
    - Process is in ready state (ready queue).
    - May be handled with aging -- dynamically increasing priority of the process.

Round-Robin

- Preemptive scheduling
- Process is assigned a time quantum/slice.
- Once time slice is completed/expired, then process is forcibly preempted and other process is scheduled.
- Min response time.

Fair-share

- CPU time is divided into epoch times.
- Each ready process gets some time share in each epoch time.
- Process is assigned a time share in proportion with its priority.
- In Linux, processes with time-sharing (TS) class have nice value. Range of nice value is -20 (highest priority) to +19 (lowest priority).
- terminal> ps -A -o pid,cls,ni,cmd

Linux scheduling classes

- Linux supports real-time (soft) and non-realtime scheduling.

- For real-time scheduling classes, high priority process is always scheduled before low priority process. In other words, high priority process never waits for a low priority process.
- Real-time scheduling classes
  - SCHED_FIFO
  - SCHED_RR
- Non Real-time scheduling classes
  - SCHED_OTHER or SCHED_NORMAL
  - SCHED_BATCH
  - SCHED_IDLE

Linux

Linux commands

- ps command
  - ps -- displays processes running in current terminal
  - ps -A -- displays all processes
  - ps aux -- displays all processes (formatted)
- kill command -- send signal to process
  - kill -sig pid
  - e.g. kill -9 7893
  - kill -l --> list of all signals
    - 9 - SIGKILL
    - 2 - SIGINT (Ctrl+C)
    - 19 - SIGSTOP (Ctrl+S)
    - 18 - SIGCONT (Ctrl+Q)
    - 15 - SIGTERM
  - pkill -sig program-name --> kill all processes of given program
    - pkill -9 java
    - pkill -kill chrome
  - When signal is sent to process, it will cause some default action.
    - Term: Terminate process (SIGINT, SIGKILL, SIGHUP, SIGALRM, ...)

        - Core: Abort process (SIGSEGV)
        - Stop: Suspend process (SIGSTOP)
        - Cont: Resume process (SIGCONT)
        - Ign: Ignore signal (SIGCHLD)
    - A program may handle the signals to implement other desired actions.
    - SIGKILL and SIGSTOP signals cannot be handled by the process.
- tr
- chmod
- chown
- ln

Multi-user related commands

- tty command
    - Display current terminal name
- who command
    - Display all users logged into the system
- w command
    - Display all users logged into the system (more details)
- whoami command
    - Display current user name
- "who am i" command
    - Display current user name and terminal.

grep

- Find a pattern in text file(s).
- Regular expressions are patterns used to match character combinations in strings.
- A regular expression pattern is composed of simple characters, or a combination of simple and special characters e.g. /abc/, /ab*c/
- Pattern is given using regex wild-card characters.
    - Basic wild-card characters
        - $ - find at the end of line.

        - ^ - find at the start of line.
        - [ ] - any single char in give range or set of chars
        - [^] - any single char not in give range or set of chars
        - . - any single character
        - zero or more occurrences of previous character
    - Extended wild-card characters
        - ? - zero or one occurrence of previous character
        - one or more occurrences of previous character
        - {n} - n occurrences of previous character
        - {,n} - max n occurrences of previous character
        - {m,} - min m occurrences of previous character
        - {m,n} - min m and max n occurrences of previous character
        - () - grouping (chars)
        - (|) - find one of the group of characters
- Regex commands
    - grep - GNU Regular Expression Parser - Basic wild-card
    - egrep - Extended Grep - Basic + Extended wild-card
    - fgrep - Fixed Grep - No wild-card
- Command syntax
    - grep "pattern" filepath
    - grep [options] "pattern" filepath
        - -c : count number of occurrences
        - -v : invert the find output
        - -i : case insensitive search
        - -w : search whole words only
        - -R : search recursively in a directory
        - -n : show line number.

Redirection

- By default every process opens 3 file descriptors
    - fd = 0 -> standard input to a program

    - fd = 1 -> standard output to a program
    - fd = 2 -> standard error to a program
- You can redirect each of these independently.
- According to direction of redirection, there are three types
- Input redirection
    - “<” is used for input redirection
- Output redirection
    - “>” is used for output redirection
- Error redirection
    - “2>” is used for error redirection

Pipe

- Using pipe, we can redirect output of any command to the input of any other command.
- Two processes are connected using pipe operator (|).
- Two processes runs simultaneously and are automatically rescheduled as data flows between them.
- If you don’t use pipes, you must use several steps to do single task.
- Syntax: command1 | command2
- E.g.
    - who | wc