Types of Command execution

Synchronous execution

- Shell waits for command to be completed.
- Default execution

Asynchronous execution

- Shell doesn't wait for command to be completed.
- To execute a command asynchronously, use "&" at the end of the command.
- terminal> gedit &

Computer structure

- CPU: Genral purpose processor for program/OS execution
- Memory: RAM
- Storage: Disk
- IO: Keyboard, Monitor
- Connected by "bus".
- Each IO device has a "dedicated" "internal" processing unit -- IO device controller.

Computer IO (Input Output)

- Synchronous IO: CPU is waiting for IO to complete.
  - Hw technique: Polling
- Asynchronous IO: CPU is not waiting for IO to complete (doing some other task)
  - Hw technique: Interrupts
  - OS maintains a device status table to keep track of IO devices (busy/idle) and processes waiting for those IO devices.

Interrupt Processing

- IO event is sensed by IO device controllers.
- It will be conveyed to CPU as a special signal - Interrupt.
- CPU pause current execution and execute interrupt handler.
- "Interrupt handler" will get address of "ISR" (from IVT) and execute ISR.
- When ISR is completed, execution resumes where it was paused.

Hardware vs Software interrupt

- Hardware -- interrupts from hardware peripherals.
- Software interrupt
  - Special instructions (Assembly/Machine level) when executed, current execution is paused, interrupt handler is executed and then the paused execution resumes.
  - Arch specific:
    - 8085/86: INT
    - ARM 7: SWI

        - ARM Cortex: SVC
    - Also called as "Trap" in few architecture.

Interrupt Controller

- Convey the interrupts from various peripherals to the CPU.
- Also manage priority of the interrupt (when multiple interrupts arrives at same time).
- e.g. 8085/86 <-- 8259, Modern x86 processors (apic), ARM-7 (VIC), ARM-CM3 (NVIC), ...

CPU Scheduling

- Modern OS are time-sharing systems i.e. CPU time is allocated to each process and after that time the next process is scheduled on the CPU.
- Timer Hardware (PIT/SysTick) is used periodically to generate the interrupt.
- When interrupt is arrived, interrupt handler is executed which in turn invokes ISR.
- Then interrupt handler invokes scheduler.
- Scheduler check if time allocated to current process is completed and if completed then decide the next process to be executed (using some scheduling algorithm).
- The selected process's execution context is then restored into CPU by the dispatcher and the next process continues to execute.
- The whole process is also referred as "Context Switch".

System Calls

- Software interrupt is used to implement OS/Kernel services.
- Functions exposed by the kernel so that user programs can access kernel functionalities, are called as "System calls".
    - e.g. Process Mgmt: create process, exit process, communication, synchronization, etc.
    - e.g. File Mgmt: create file, write file, read file, close file, etc.
    - e.g. Memory Mgmt: alloc memory, release memory, etc.
    - e.g. CPU Scheduling: Change process priroty, change process CPU affinity, etc.
- System calls are specific to the OS:
    - UNIX: 64 syscalls e.g. fork(), ..
    - Linux: 300+ syscalls e.g. fork(), clone(), ...
    - Windows: 3000+ syscalls e.g. CreateProcess(), ...

Bash Shell Scripts

Introduction

- Shell script is collection of commands along with programming syntax.
- Different shells have different syntax.
- Bash shell script.
- Shebang line should contain path of shell program that will execute the script.

Applications

- Administrative tasks e.g. User creations, Troubleshooting, Router/Firewall config on commandline, etc.
- Automation of tasks e.g. Cron jobs, Scripts for building softwares, etc.

* Application installers

Limitations

* Slow execution -- Interpreted. Cannot be used for performance centric applications.