Unix Internals
OS Architecture

Khader Shaik

Unix OS - Agenda

- Boot Process
- Shutdown Process
- Process Management
- Memory Management
- File System
- Networking

PROM – Programmable ROM

- Small non-volatile memory loaded with vendor specific software
- Performs some basic checks (POST) and finds the boot device; runs the boot program
  - Power-on Self Test (POST)
- Usually Boot program is stored on Block 0 (boot sector) of a Boot disk
- Boot program reads and loads the kernel program
- System can be booted from any other devices as well if you have access to ROM
  - > ***boot cdrom*** (on Sun)
- More details are Vendor specific

Boot Process - Steps

- Kernel checks the Memory Availability
  - For eg: on Solaris kernel program is located under - /kernel/unix
- Probes & Configures hardware devices
- Updates the entries in ***/dev*** directory
  - /dev directory holds the information on all hardware devices installed
- Sets up all RAM tables to hold the processes information, memory allocation and open files etc
- At the of the booting KERNEL creates the *init* process

Dummy Processes

- Process that cannot be killed is known as DUMMY process
- Used for crucial system functions
- Created by Kernel during booting process
  - Part of the kernel
- Some of the samples are:
  - swapping services (xsched)
  - virtual memory paging services (pageout/vhand)
  - periodic buffer flushing service (bdflush)
- To see these process run command > *ps -ef*

OS Run Levels

- OS runs in different levels indicating a specific mode
- *init* command changes the run level of OS (init process is different from init command)
- Different levels have the different limited set of processes running
  - ***/etc/inittab*** – configuration file contains this info
- Different sample levels

0 – Halt 1 – single user mode 2 – Multi-user without NFS
3 – Full multi-user 4 – unused 5 – X11 graphical mode
6 – Reboot etc.

Process Management

- Process Hierarchy
- Process Creations
- Process Management
- Process Status
- Communicating with Processes

Process Hierarchy

- KERNEL process is created as part of the booting
- KERNEL first loads the ***init*** process
- All other processes are created by ***init***
- Each process is given an unique process id (PID)
  - PID of ***init*** is 1
  - PID of kernel is 0 (zero)
- Each process is associated with its creator (parent) – identified by the Parent Process ID (PPID)
- *init* process is mother of all processes

Process Hierarchy ..cont

- When parent process of any process dies, it will be linked to ***init*** process as its parent
- Each process runs with a specific priority (PRI)
- For Eg:
  - Solaris Priorities vary from -20 to +20;
  - '-20' being highest and '+20' being the lowest
- 'ps' command lists the current processes running
  - 'ps –ax'

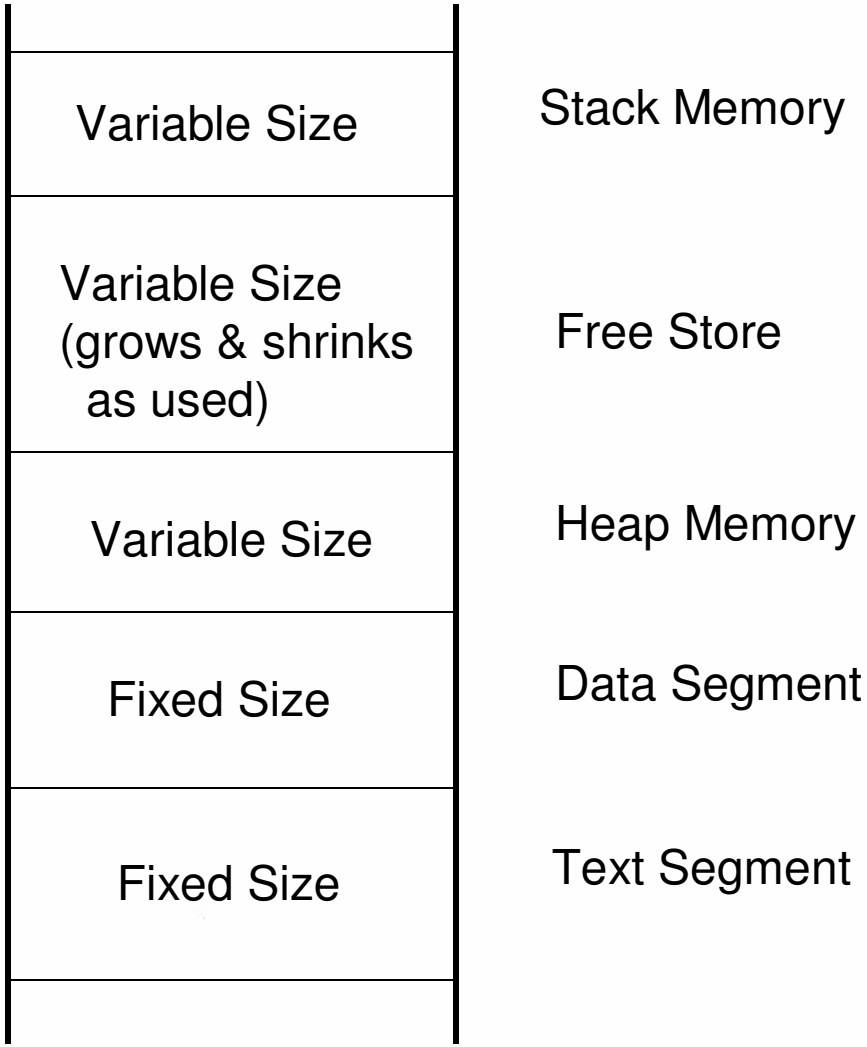
Process Creation

- New processes are created using either ***fork*** or ***exec*** command
- ***fork*** creates an independent process
  - Except kernel everything else is created using *fork*
- ***exec*** creates a new process as a sub-process (thread) of the calling program
  - It shares the memory and other resources of the parent process
- Eg: a shell script with ***find*** command in it
  - Shell creates the independent (fork) process to execute the Shell script
  - Script process creates the sub-process (using *exec*) to execute the ***find*** command

Process Creation ..

- Kernel allocates the memory to the new process known as ADDRESS SPACE
- Address Space contains four main segments
  - Text – stores the program instructions
  - Data – contains program variables (initialized) (dynamically grows?)
  - BSS/Heap Memory - contains un-initialized program variables (dynamically shrinking as used)
  - Free Store – unused memory; used as programs allocates
  - Stack – stores local variables and function parameters (dynamically grows)

Process Address Space (each process)



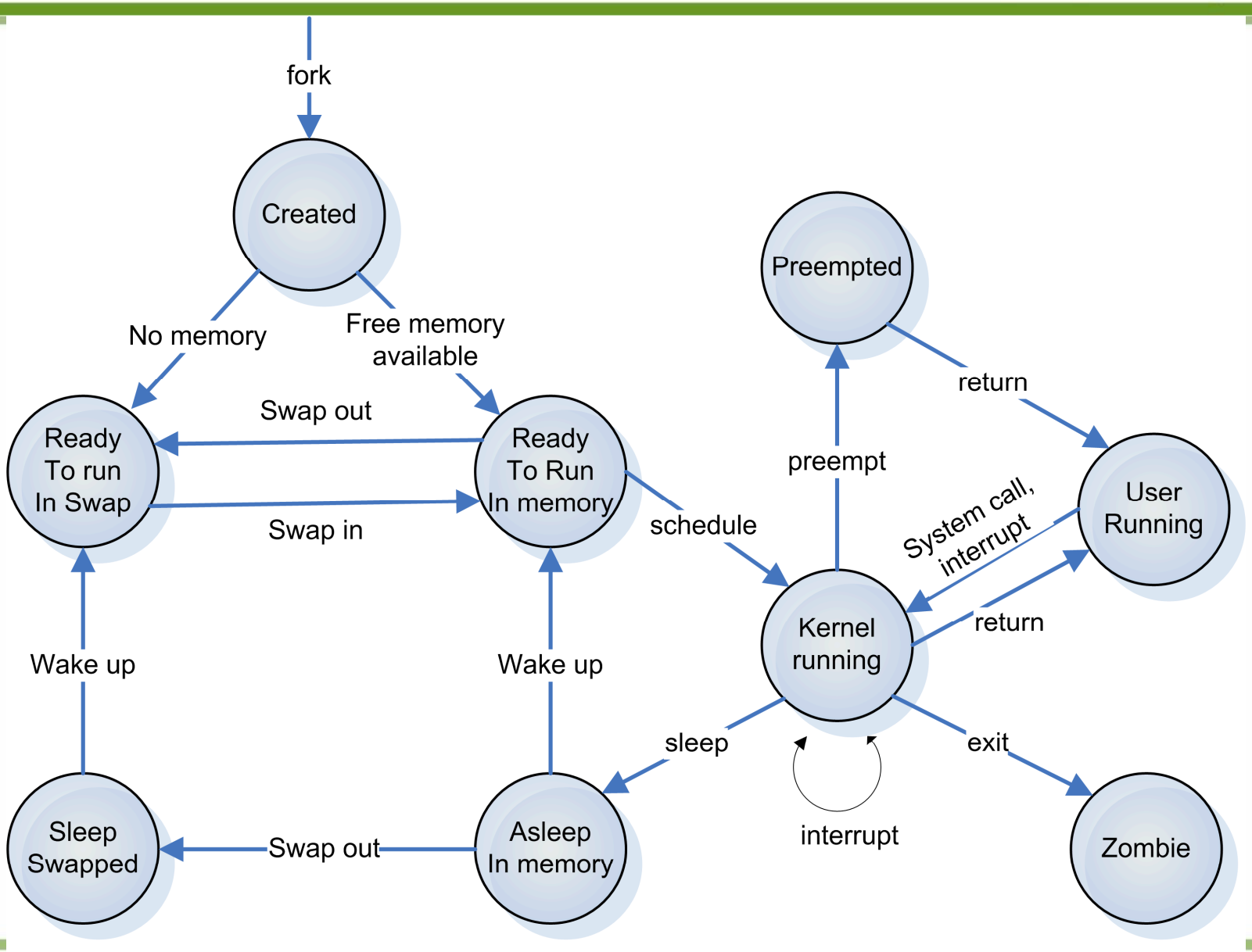
Process Execution Modes

- Kernel Mode
  - When process is executing kernel instructions is known as in KERNEL MODE
  - Control transfers to the Kernel and kernel carries out the instructions on behalf of the User process
  - During this mode, process can access entire Address Space of any process
  - Eg: User process is making a System call, interrupt, generating exception etc
- User Mode
  - Process created by user and executing in CPU is known as in USER MODE
  - It can access only its Address space and can't access any other user's space

Various Process States

- Created
- Ready to run in Memory
- Ready to run in Swap
- Asleep in memory
- Sleep, swapped
- User Running
- Kernel Running
- Preempted
- Zombie

Process Life Cycle



Process State Diagram

Process Interruptions – Two Types

- **Interrupt -** Caused by some event that is external to and asynchronous to the currently running process
  - Eg: Completion of IO.
- **Trap -** Error or exception condition generated within the currently running process.
  - Eg: illegal access to a file, arithmetic exception.
- ?(supervisor call) : explicit interruption

Common SIGNALS

- SIGHUP - Hang-up
- SIGINT- Interrupt
- SIGQIT - Quit
- SIGINS - Illegal Instruction
- SIGTRAP - Trace Trap
- SIGKILL - Kill
- SIGSYS - Bad argument to system call
- SIGPIPE - Write on pipe with no one to read it
- SIGTERM - Software termination signal from kill
- SIGSTOP - Stop signal
- For more info - /usr/include/sys/signal.h