**I**

**UNIX environment**

1. What does the bin directory contains? What does the dev directory contains?

Bin Directory - Useful commands that can be used by both system administrator as well as non-privileged user. Example, ls and sleep

Dev Directory – contains the special device files for all the devices. The device files are created during installation later with dev/makedev. Example, Bus and CPU

1. Discuss the creation of a process from the point of view of address space.

**Process Creation**

In terms of execution:

* The parent continues to execute concurrently with its children
* The parent waits until some or all of its children have terminated.

In terms of resource utilization:

* The child may be able to obtain the resources directly from the OS
* The child may be constrained to a subset of the resources of the parent process. (good for deadlock handling)

In terms of the address space:

* The child is a duplicate of the parent. (same address space & variables with same values)
* The child is a separate program (different address space).

1. What is the name of system call (command) that can overwrite the default address? What is the meaning of the possible arguments that this commands takes?

Exec() – it can take different parameters. Example, file name and path to the file.

1. What happens to a child process if its parent is terminated?

If parent is terminated, INIT process adopts the children of terminated process.

1. What are the return values for the fork() command and what are their meaning?

**Fork() Returned Value**

-1 or < 0 if creation failed

0 returned value to the child

1 – nnn The PID of the child returned to the parent process

1. Define a zombie process.

**Zombie State** – is between the time a child terminated until its parent detect the termination by calling wait(). PCB of the child remains in main memory

**II**

1. Explain the main difference between a mode switch and a full context switch. Give example for each type of switch.

**Mode Switch**

1. It saves the context of the processor
2. It sets the PC to the starting address of an OS program, the interrupt handler routine. The Interrupt Handler checks the cause of the interrupt and it might service the interrupt. One of the main tasks of the Interrupt Handler Routine is to protect the PCB of the process. Usually it is the only process that modifies information inside the PCB. (In more complicated situation, the Interrupt Handler will call a specific Interrupt Service Routine that will service the Interrupt).
3. Meanwhile the processor switched from the user mode to the system mode.

**Full Context Switch**

1. Save the context of the processor.
2. Update the PCB of the process that is currently in the Running state.
3. Move the PCB of this process to the appropriate queue
4. Select another process for execution.
5. Update the PCB of the new process.
6. Update memory management data structures.
7. Restore the context of the processor to that which existed at the time the selected process was last switched out of the Running state.

Process Switch times are pure overhead and are highly dependent on hardware support.

**Process Operations**

**Activate** – restart the process from the point at which it was suspended. The process is moved from the suspended (swapped-out) in secondary memory to the corresponding active state.

**Swapped (suspended in secondary memory**) – it is often an operation performed by the OS to control the number of processes residing in the main memory, the degree of multiprogramming. Most of the time the suspension last only brief periods of time. A process remains suspended until ANOTHER process activates it. Swapping is a high-priority operation.

**Release** – is required to move the process from either of the blocked state to the corresponding ready state.

**Process States**

**New** – a process that just has been created but has not been yet admitted to the pool of executable processes.

**Ready swapped** – the process is in secondary memory, but is available for execution as soon as it is loaded into main memory.

**Ready(active)** – the process is in main memory and available for execution.

**Running** – the process is currently being executed.

**Blocked** – the process is in main memory and awaiting an event. The process doesn’t have all the resources it needs.

**Block Swapped** – the process is in secondary memory and it may wait for an event.

**Terminated** - the process has finished execution.

**UNIX Process Termination**

A process may terminate by using the exit system call, and its parent process may wait for that event. If the parent terminates, all its children have assigned as their new parent the init process. Echo$? displays the exit code of the last process.

**MS-DOS Process Termination**

When a process finishes executing, asks the OS to delete it by using the exit system call. The process may return data to its parent via the wait (provides the process id) system call.All the resources of the process are deallocated. A process can cause the termination of another process using abort system call. Usually only the parent of the process that is to be terminated can invoke such system call. On PC a user may quit its application. –Everything results in a system call to the OS to terminate the requesting process.

A parent may terminate the execution of one of its children because

* The child exceeded the usage of some resources.
* The task assigned to the child is no longer required
* The parent is exiting and the OS does not allow a child to continue if its parent terminates.

**Processes unaware of each other**: Independent processes that are not intended to work together. The O.S. needs to be concerned about the competition for resources. The best example is the multiprogramming of multiple independent processes.

**Potential Control Problems**: Mutual Exclusion, Deadlock, Starvation.

**Processes indirectly aware of each other:** processes those are not necessarily aware of each other by name but share access to some object, such as an I/O buffer. Such processes need cooperation in sharing the common object. Multiple processes may have access to shared variables, files or databases.

**Potential Control Problems:** Mutual Exclusion, Deadlock, Starvation, Data Coherence. (Producer/Consumer, Reader/Writer problem)

**Processes Directly aware of each other:** processes able to communicate with each other by name. The communication provides a way to synchronize or coordinate the various activities.

**Potential Control Problems:** Deadlock, Starvation.