# Section 3: Process Description and Control

## Question 1

Answers to this question are found in the textbook.

1. Explain the meaning of a process’s instruction trace.
2. Explain the instruction trace presented in Figure 3.4 in the book.   
   How many processes are represented in this figure?   
   What is meant by Time-Out and by I/O Request?
3. Explain the role of the dispatcher service in all of this. How is the dispatcher represented in Figure 3.4?

### Answer

1. The instruction trace is a list (or trace) of the instruction address that a process executes over some given time range while in the ‘Running’ state.
2. Figure 3.4 illustrates the execution of three processes interleaved with the dispatcher’s execution. The execution of a process divided into slices and every eligible process is to be given a turn executing. The process’s execution can be interrupted for two reasons in this example. Either the process executes a Blocking I/O Request (Blocking SYSCALL) or it executes for the full duration of its time-slice and is preempted by time-out interrupt.
3. The dispatcher is the OS mechanism that selects, installs, and allows to execute a “ready to execute” process from the Read Process Queue. The dispatcher’s execution is represented by the gray shaded traces which execute the same instructions each time.

## Question 2

1. What is a Process Image?
2. In what two places does a process image reside?
3. What are the five generic steps involved with building a new process image?

### Answer

1. The process image is the region of memory (disk or main memory) containing the instructions / data / stack of the program being executed. Note that in the model presented in the text, the image also contains the Process Control Block. In this class, the PCB resides in the Kernel for security purposes.
2. The process image resides on the system drive (as a file) and in main memory. The image is initially created on the drive and copied into main memory for execution.
3. The five generic steps are:
   1. Allocate space for the new image on the system (swap) drive. Copy text and data segments from the executable file to the new image.
   2. Create the Process Control Block and other kernel data structures needed by the OS to maintain the process’s execution.
   3. Assign a unique process ID (PID) to the new process.
   4. Copy the image from the system drive into main memory.
   5. Put the new Process in the Ready state and place in the Ready FIFO Queue.

## Question 3

Using Figure 3.6, describe the process states and conditions that cause a process to transition from one state to another i.e. 1) The transition from New to Ready. 2) Ready to Running 3) Running to Ready 4) Running to Blocked 5) Blocked to Ready 6) Running to Exit.

### Answer

New to Ready: When a new process has been fully installed by the OS i.e. its Process Control Block has been created, its process image has been created, and it is ready to be installed in the ready queue for its turn at execution.

Ready to Running: The dispatcher selects a ready process from the Ready Process FIFO. The process’s Processor Context is installed on the processor and the process begins executing its current slice.

Running to Ready: The process has completed it slice (time-out). Is processor context is saved and it is moved to the back of the FIFO waiting for its turn to be re-scheduled.

Running to Blocked: The process has executed a blocking system call (e.g. I/O request) and its execution is blocked while it waits for the associated event (interrupt) signaling that the request has completed and the process can resume execution.

Blocked to Ready: The blocked process’s event has occurred and it is once again eligible for execution. It is moved from the blocked list to the ready queue.

Running to Exit: The process has been released by the OS for one of several reasons given in the book and slides. The process control block is release and system resources allocated to the process are also released.

## Question 4

Describe the difference between the monolithic kernel architecture and the process-based kernel architecture.

Include a description about how user processes access kernel services in each model.

### Answer

The monolithic kernel architecture implements the kernel as a single (monolithic) block of executable instructions that is loaded and executed at system boot. The kernel instructions provide the facilities needed to create and maintain processes, device interrupt handlers, and individual syscall instructions (e.g. read(), open(), malloc(), etc.). Also included is the syscall handling mechanism used to transfer control from a process to code residing in the kernel program.

The process-based kernel architecture (also known as a Microkernel architecture) implements its services as ‘kernel processes’ (see Slide 39). Kernel Processes are processes separate from the MicroKernel, but run in Kernel Mode as these processes can access individual devices. Both user programs and kernel services are implemented as individual processes. Processes (user and kernel) are interconnected by a relatively simple service layer that provides process management (creation and switching) and a fast Inter-Process Communication. The later allows user processes to make service requests (syscalls) to kernel processes.

## Question 5

Describe what it means to “Suspend a Process”.

Why would the system operator choose to suspend a process?

What is the benefit of suspending a process?

### Answer

1. A suspended process is one that has been set into a ‘suspended’ state and while in the suspended state, will not be scheduled for execution (i.e. made Ready).
2. Suspension is useful if the system finds itself over-capacity. That is, if there is a shortage of physical memory, I/O capacity, or other limited resource such that the system begins thrashing and no progress is being made by any of the executing processes.
3. The benefit of process suspension is that a process can be temporarily halted (i.e. suspended from execution) freeing memory, processor, and other resources (I/O capacity and others) for the remaining executing processes. Because the process is only suspended (and not terminated), a suspended process can later be un-suspended and continue execution.

## Question 6

1. What are the two system calls used to create and launch a new executable process in the Linux (Unix) operating system.
2. Describe how a parent process uses these two calls to create a new child process from a given executable file.
3. What is meant by copy on write?

You can use the book “Linux Kernel Development”, Section 3 found in the Supplemental Materials eLearning folder.

### Answer

The two system calls are fork() and exec().

1. The fork() system call logically duplicates the process image that makes the call. The new process is a logical clone of the original with the exception of the process ID and parent process ID.
2. The exec(‘exec\_file’) system call replaces the process image of the process making the call with a new image obtained from an executable file that was passed to the syscall.

So creating a new executable process involves 1) having the parent process call fork() to create a new distinct child process and 2) having the child process call exec() to replace the parent’s image with the image of the desired executable.

Copy on Write is the technique the Linux kernel uses to avoid physically copying the entire parent process image into the new child process. This copy would be especially wasteful considering that under most circumstances the child process is going to be immediately replaced with the new image of the desired executable.

## Question 7

Does every mode switch (i.e. system call) result in a blocked process? Explain why or why not.

### Answer

No. The process may switch to kernel mode in response to a system call that involves no I/O or other blocking operation. For example a call to malloc() or free() to allocated or free memory can be executed by the process causing a mode switch. However, these calls only update internal OS tables and so control can be returned to the calling process without the process entering the ‘blocked’ state.

## Question 8

Describe the four mechanisms that are used to preempt a process’s execution. Two of the four mechanisms are derived from interrupts (two types of hardware interrupts) and the two others as presented in the slides.

### Answer

The four mechanisms are:

1. Device Interrupts: A hardware signal from a system controller that signals that the controller will soon need servicing.
2. Timer Interrupts: The interrupt generated by the hardware timer that interrupts the currently executing process so that the dispatcher can gain control and perform a context switch. Although the timer interrupt basically a hardware interrupt, its purpose is fundamentally different from a the interrupts generated by system controllers.
3. Blocking System Calls (SYSCALL): A request by the process to the OS for some service. Implemented by a software interrupt instruction (INT).
4. Signals: Also known as Traps. These are events generated by the OS that ‘signal’ the occurrence of something of interest. For example, division by zero errors or memory bounds violations are examples of signals that would cause the termination of the executing process.

## Question 9

1. Describe the meaning of “Context Switch”.
2. What two events trigger a context switch?
3. What OS component is responsible for performing a context switch?
4. Describe a context switch’s 6 steps as found in the slides.

### Answer

1. The context switch is the process of halting / blocking the execution of the currently “running” process and the installation of the next ‘ready’ process for execution (running).
2. A context switch is triggered by either a timer interrupt or by the running process executing a blocking system call.
3. The Dispatcher is activated and performs the steps shown below.
4. The six steps are:
   1. Save the information needed to restore the process the currently running process.
   2. Change the running process’s state to Ready or to Blocked.
   3. The dispatcher selects the next process to be scheduled from the READY queue.
   4. Change the selected process’s state to Running.
   5. Restore the selected process’s Processor Context from its PCB.
   6. The running process resumes execution from where it was last interrupted.

## Question 10

1. Describe the significance of a Process’s Priority.
2. How does a process’s priority affect the selection of the next ready process from the dispatch FIFO?

### Answer

1. A process is assigned a ‘priority’ which describes its ‘importance’ in the running of the system. Some processes are more important and so have a higher priority than other less important processes. On a desktop system, the process running the GUI (windows) runs at a higher priority than an application e.g. Word. This is because the user’s experience will be poorer if the GUI seems sluggish or not responsive.
2. The priority affects what processes are dispatched during a context switch. We described the ready queue as FIFO. This is true if all the ready processes have the same priority. However, “ready” Pi has a higher priority than other ready processes, then it will be placed at the head of the FIFO so that the dispatcher selects before the others for execution.

## Also know for exam:

* The PCB and categories of information maintained. This topic was introduced in Section 2 and in more detail in Section 3.