Chapter 3

Multiple Choice Questions

1. Which of the following contains dynamically allocated data during program run time?
A) text section
B) data section
C) heap section
D) stack section

Ans: C
Feedback: 3.1.1
Difficulty: Easy

2. Which of the following contains the executable code?
A) text section
B) data section
C) heap section
D) stack section

Ans: A
Feedback: 3.1.1
Difficulty: Easy

3. Which of the following contains global data?
A) text section
B) data section
C) heap section
D) stack section

Ans: B
Feedback: 3.1.1
Difficulty: Easy

4. Which of the following contains memory allocated by *malloc()*?
A) text section
B) data section
C) heap section
D) stack section

Ans: C
Feedback: 3.1.1
Difficulty: Easy

5. Which of the following memory space dynamically grows from high memory to low memory?
A) text section
B) data section
C) heap section
D) stack section

Ans: D
Feedback: 3.1.1
Difficulty: Easy

6. Which of the following process state will be switched from “running” state when an interrupt occurs?
A) ready
B) terminated
C) waiting
D) new

Ans: A
Feedback: 3.1.2
Difficulty: Easy

7. Which of the following process state will be switched from “running” state when an I/O event occurs?
A) ready
B) terminated
C) waiting
D) new

Ans: C
Feedback: 3.1.2
Difficulty: Easy

8. Which of the following process state will be switched from “ready” state?
A) ready
B) terminated
C) waiting
D) running

Ans: D
Feedback: 3.1.2
Difficulty: Easy

9. Which of the following data structure is used to manage the processes in Linux?
A) tree
B) linked list
C) hash map
D) stack

Ans: B
Feedback: 3.2
Difficulty: Easy

10. Which of the following structures defines the process control block for Linux system?
A) task_struct
B) files_struct
C) mm_struct
D) schedu_entity

Ans: A
Feedback: 3.2
Difficulty: Medium

11. What is the degree of multiprogramming for a single-processor system?
A) 0
B) 1
C) 2
D) 3

Ans: 1
Feedback: 3.2
Difficulty: Easy

12. Which of the following is true?
A) An I/O-bound process is one that spends more of its time doing I/O than it spends doing computations
B) An I/O-bound process is one that spends more of its time doing computations than it spends doing I/O
C) Both A and B
D) None of the above

Ans: A
Feedback: 3.2
Difficulty: Easy

13. Which of the following cases could force a process removed from the CPU?
A) I/O request
B) fork a child
C) interrupt or time slice expired
D) all of the above

Ans: D
Feedback: 3.2.1
Difficulty: Medium

14. The list of processes waiting to execute on a CPU is called a(n) ____.
A) standby queue
B) device queue
C) ready queue
D) interrupt queue

Ans: C
Feedback: 3.2.2
Difficulty: Easy

15. Which of the following selects from among the processes that are in the ready queue to execute and allocate the CPU to one of them?
A) CPU scheduler
B) context switch
C) swapping
D) job scheduler

Ans: A
Feedback: 3.2.2
Difficulty: Easy

16. If process P_0 is switched to process P_1, state for P_0 will be saved into ____, and state from ___ will be reloaded?

A) PCB_0, PCB_0
B) PCB_0, PCB_1
C) PCB_1, PCB_0
D) PCB_1, PCB_1

Ans: B
Feedback: 3.2.2
Difficulty: Medium

17. The ________ application is the application appearing on the display screen of a mobile device.

A) main
B) background
C) display
D) foreground

Ans: D
Feedback: 3.2.3
Difficulty: Easy

18. What is done by command "*spy -el*" on Unix and Linux systems?

A) list complete information for all processes currently active in the system
B) list complete information for all processes currently running in background
C) list complete information for all files currently open in the system
D) list complete information for all folders currently open in the system

Ans: A
Feedback: 3.3.1
Difficulty: Easy

19. Which of the following system calls is used to have a new program loaded into the new process's memory space?

A) fork()
B) exec()
C) wait()
D) exit()

Ans: B
Feedback: 3.3.1
Difficulty: Easy

20. Which of the following system calls is used to let the parent process create a child process?
A) abort();
B) wait();
C) fork();
D) exec().

Ans: C
Feedback: 3.3.1
Difficulty: Easy

21. Which of the following IPC mechanism is easier to implement in a distributed system?
A) shared memory
B) message passing
C) socket communication
D) ordinary pipe

Ans: B
Feedback: 3.4
Difficulty: Easy

22. Which of the following is not a process type in the Chrome browser?
A) Plug-in
B) Renderer
C) Sandbox
D) Browser

Ans: C
Feedback: 3.4
Difficulty: Medium

23. Assume the shared buffer is implemented as a circular array with two logical pointers: *in* and *out*. The variable *in* points to the next free position in the buffer; *out* points to the first full position in the buffer. Which of the following is true?
A) The buffer is empty when in == out; the buffer is full when ((in + 1) % BUFFER SIZE) == out;
B) The buffer is full when in == out; the buffer is empty when ((in + 1) % BUFFER SIZE) == out;
C) All the elements of the buffer can be used;
D) Both A and C

Ans: A
Feedback: 3.5
Difficulty: Medium

24. Which of the following is true in a Mach operating system?
A) All messages have the same priority.
B) Multiple messages from the same sender are guaranteed an absolute ordering.
C) The sending thread must return immediately if a mailbox is full.
D) It is not designed for distributed systems.

Ans: A
Feedback: 3.7.2
Difficulty: Medium

25. Child processes inherit UNIX ordinary pipes from their parent process because:
A) The pipe is part of the code and children inherit code from their parents.
B) A pipe is treated as a file descriptor and child processes inherit open file descriptors from their parents.
C) The STARTUPINFO structure establishes this sharing.
D) All IPC facilities are shared between the parent and child processes.

Ans: B
Feedback: 3.7.4.1
Difficulty: Medium

Short Answer Questions

1. Name and describe the different states that a process can exist in at any given time.

Ans: The possible states of a process are: new, running, waiting, ready, and terminated. The

process is created while in the new state. In the running or waiting state, the process is executing or waiting for an event to occur, respectively. The ready state occurs when the process is ready and waiting to be assigned to a processor and should not be confused with the waiting state mentioned earlier. After the process is finished executing its code, it enters the termination state.
Feedback: 3.1.2
Difficulty: Medium

2. Explain the main differences between a short-term and long-term scheduler.

Ans: The primary distinction between the two schedulers lies in the frequency of execution. The short-term scheduler is designed to frequently select a new process for the CPU, at least once every 100 milliseconds. Because of the short time between executions, the short-term scheduler must be fast. The long-term scheduler executes much less frequently; minutes may separate the creation of one new process and the next. The long-term scheduler controls the degree of multiprogramming. Because of the longer interval between executions, the long-term scheduler can afford to take more time to decide which process should be selected for execution.
Feedback: 3.2.2
Difficulty: Medium

3. Explain the difference between an I/O-bound process and a CPU-bound process.

Ans: The differences between the two types of processes stem from the number of I/O requests that the process generates. An I/O-bound process spends more of its time seeking I/O operations than doing computational work. The CPU-bound process infrequently requests I/O operations and spends more of its time performing computational work.
Feedback: 3.2.2
Difficulty: Medium

4. Explain the concept of a context switch.

Ans: Whenever the CPU starts executing a new process, the old process's state must be preserved. The context of a process is represented by its process control block. Switching the CPU to another process requires performing a state save of the current process and a state restore of a different process. This task is known as a context switch. When a context switch occurs, the kernel saves the context of the old process in its PCB and loads the saves context of the new process scheduled to run.
Feedback: 3.2.3
Difficulty: Easy

5. Ordinarily the exec() system call follows the fork(). Explain what would happen if a programmer were to inadvertently place the call to exec() before the call to fork().

Ans: Because exec() overwrites the process, we would never reach the call to fork() and hence, no new processes would be created. Rather, the program specified in the parameter to exec() would be run instead.
Feedback: 3.3.1
Difficulty: Medium

6. Explain the fundamental differences between the UNIX fork() and Windows CreateProcess() functions.

Ans: Each function is used to create a child process. However, fork() has no parameters; CreateProcess() has ten. Furthermore, whereas the child process created with fork() inherits a copy of the address space of its parent, the CreateProcess() function requires specifying the address space of the child process.
Feedback: 3.3.1
Difficulty: Medium

7. Describe how UNIX and Linux manage orphan processes.

Ans: If a parent terminates without first calling wait(), its children are considered orphan processes. Linux and UNIX assign the init process as the new parent of orphan processes and init periodically calls wait() which allows any resources allocated to terminated processes to be reclaimed by the operating system.
Feedback: 3.3.2
Difficulty: Medium

8. Explain why Google Chrome uses multiple processes.

Ans: Each website opens up in a separate tab and is represented with a separate renderer process. If that webpage were to crash, only the process representing that the tab would be affected, all other sites (represented as separate tabs/processes) would be unaffected.
Feedback: 3.4
Difficulty: Medium

9. Name the three types of sockets used in Java and the classes that implement them.

Ans: Connection-oriented (TCP) sockets are implemented with the Socket class. Connectionless (UDP) sockets use the DatagramSocket class. Finally, the MulticastSocket class is a subclass of the DatagramSocket class. A multicast socket allows data to be sent to multiple

recipients.
Feedback: 3.8.1
Difficulty: Medium

10. What is a loopback and when is it used?

Ans: A loopback is a special IP address: 127.0.0.1. When a computer refers to IP address 127.0.0.1, it is referring to itself. When using sockets for client/server communication, this mechanism allows a client and server on the same host to communicate using the TCP/IP protocol.
Feedback: 3.8.1
Difficulty: Medium

11. Explain the purpose of external data representation (XDR).

Ans: Data can be represented differently on different machine architectures (e.g., *little-endian* vs. *big-endian*). XDR represents data independently of machine architecture. XDR is used when transmitting data between different machines using an RPC (remote procedure call).
Feedback: 3.8.2
Difficulty: Hard

12. Explain the term marshalling.

Ans: Marshalling involves the packaging of parameters into a form that can be transmitted over the network. When the client invokes a remote procedure, the RPC system calls the appropriate stub, passing it the parameters provided to the remote procedure. This stub locates the port on the server and marshals the parameters. If necessary, return values are passed back to the client using the same technique.
Feedback: 3.8.2
Difficulty: Medium

13. Explain the terms "at most once" and "exactly once" and indicate how they relate to remote procedure calls.

Ans: Because a remote procedure call can fail in any number of ways, it is important to be able to handle such errors in the messaging system. The term "at most once" refers to ensuring that the server processes a particular message sent by the client only once and not multiple times. This is implemented by merely checking the timestamp of the message. The term "exactly once"

refers to making sure that the message is executed on the server once and only once so that there is a guarantee that the server received and processed the message.
Feedback: 3.8.2
Difficulty: Hard

14. Describe two approaches to the binding of client and server ports during RPC calls.

Ans: First, the binding information may be predetermined, in the form of fixed port addresses. At compile time, an RPC call has a fixed port number associated with it. Second, binding can be done dynamically by a rendezvous mechanism. Typically, an operating system provides a rendezvous daemon on a fixed RPC port. A client then sends a message containing the name of the RPC to the rendezvous daemon requesting the port address of the RPC it needs to execute. The port number is returned, and the RPC calls can be sent to that port until the process terminates (or the server crashes).
Feedback: 3.8.2
Difficulty: Hard

True/False Questions

1. The difference between a program and a process is that a program is an active entity while a process is a passive entity.

Ans: False
Feedback: 3.1.1
Difficulty: Easy

2. For a single-processor system, there will never be more than one process in the Running state.

Ans: True
Feedback: 3.1.2
Difficulty: Easy

3. The iOS mobile operating system only supports a limited form of multitasking.

Ans: True
Feedback: 3.2.3
Difficulty: Medium

4. The exec() system call creates a new process.

Ans: False
Feedback: 3.3.1
Difficulty: Easy

5. init is the very first process for a typical Linux system.

Ans: True
Feedback: 3.3.1
Difficulty: Easy

6. All processes in UNIX first translate to a zombie process upon termination.

Ans: True
Feedback: 3.3.2
Difficulty: Hard

7. Background processes are not apparent to users.

Ans: True
Feedback: 3.3.2
Difficulty: Easy

8. All access to POSIX shared memory requires a system call.

Ans: False
Feedback: 3.7.1
Difficulty: Easy

9. The Mach operating system treats system calls with message passing.

Ans: True
Feedback: 3.7.2
Difficulty: Medium

10. Local Procedure Calls in Windows XP are similar to Remote Procedure Calls.

Ans: True
Feedback: 3.7.3
Difficulty: Easy

11. Using a section object to pass messages over a connection port avoids data copying.

Ans: True
Feedback: 3.7.3
Difficulty: Medium

12. Sockets are considered a high-level communications scheme.

Ans: False
Feedback: 3.8.1
Difficulty: Medium

13. Ordinary pipes in UNIX require a parent-child relationship between the communicating processes.

Ans: True
Feedback: 3.7.4.1
Difficulty: Easy

14. Ordinary pipes in Windows require a parent-child relationship between the communicating processes.

Ans: True
Feedback: 3.7.4.1
Difficulty: Easy

15. Ordinary pipes are unidirectional, allowing only one-way communication.

Ans: True
Feedback: 3.7.4.1
Difficulty: Easy

16. The communication with named pipes requires parent-child relationship.

Ans: False
Feedback: 3.7.4.2
Difficulty: Easy