King Saud University
Department of Computer Science
CSC227: Operating Systems
Tutorial No. 3

Q 1) Discuss the different states that a process can exist in at any given time.

Answer

The possible states of a process are: new, running, waiting, ready, and terminated. The process is created while it is 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.

Q 2) Describe the actions taken by a kernel to context-switch between processes.

Answer

In general, the operating system must save the state of the currently running process and restore the state of the process scheduled to be run next. Saving the state of a process typically includes the values of all the CPU registers in addition to memory allocation. Context switches must also perform many architecture-specific operations, including flushing data and instruction caches.

Q 3) What are the differences between a short-term and long-term scheduler?

Answer

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.

Q 4) Discuss in details the producer/consumer problem.

Answer

In the producer/consumer problem, a producer creates items and a consumer uses them. The items are stored in a shared buffer, which can be infinite or of a limited size. The producer and consumer must synchronize on the buffer contents so that items are not lost or consumed more than once. If the buffer is empty, the consumer must wait for the producer to create a new item. If a finite buffer is full, the producer must wait for the consumer to use an item.

Q 5) List three different situations lead to the following direct transitions between states: Running state to ready state; waiting state to ready state; running state to waiting state

Answer

- Running state to ready state
  Answer: Interrupt, time-out (end of time slice), arrival of a higher priority process, sleep system call.
- Waiting state to ready state

Answer: End of I/O, parent becomes ready after child complete execution

- Running state to waiting state

Answer: Request for I/O, process forks a child, wait for an event, wait system call.

Q 6) Describe the actions taken when a user process starts to execute a write() system call.

Answer

1) The write() call is a privileged instruction that traps to the kernel.
2) The kernel saves the user process context.
3) The kernel sets up registers in the I/O controller and transfers the data to be written to the controller.
4) The kernel selects a (different) ready process and switches context to that process.
5) When output is complete, the I/O device generates an interrupt.
6) The interrupt service routine saves the current CPU context, does some accounting, and moves the writing user process into the ready queue.
7) The kernel either 1) switches context to the original process executing the system call, or 2) returns control to the second process, or 3) selects and switches context to an entirely different process.

Q 7) How many times the message will be printed?

```
main()
{
fork();
printf("Hello world");
}
```

Answer: Two times. Child is created at fork() and every line after fork will be executed twice once by parent (i.e. printf("Hello world");) and once by child (again printf("Hello world");)

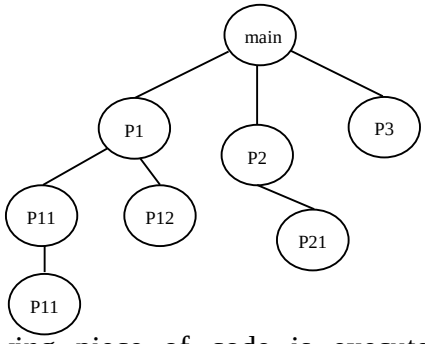
Q 8) Consider the following C language program, what are the possible outputs?

```
int num;
main() {
num = 1;
fork();
num = num + 1;
printf(num);
}
```

Answer: It will print value of `num` two times one from parent and one from child.

Q 9) How many times does the program below print Hello?

```
main() {
fork();
fork();
fork();
printf( "Hello\n" );
}
```

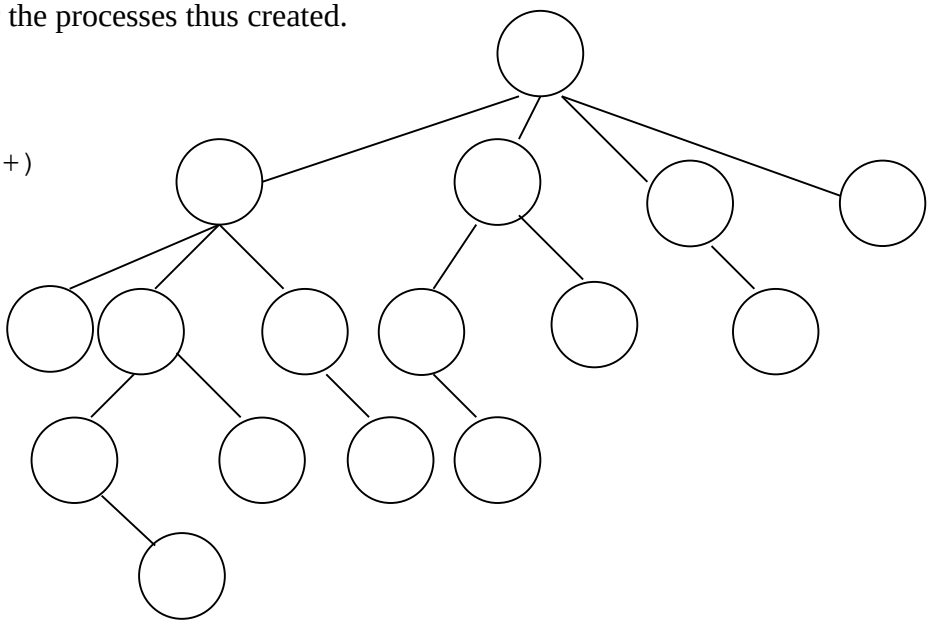


Answer: 8 times.

Q 10) How many processes are created when the following piece of code is executed? Draw the process tree for the processes thus created.

```
main() {
int i;

for (i=0; i<4; i++)
fork();
return 1;
}
```



Answer:15 + main

Q 11) Using the program shown, explain what the output will be at Line A.

```
int value = 5;
main(){
pid_t pid;
pid = fork();
if (pid == 0) { /* child process */
value += 15;
return 0;
}
else if (pid > 0) { /* parent process */
wait(NULL);
printf("PARENT: value = %d",value); /* LINE A */
return 0;
}
}
```

Answer: The result is still 5 as the child updates its copy of value. When control returns to the parent, its value remains at 5.

Q 12) Which message will show up on the screen?

```
main(){
int pid;
printf("Before fork()\n");
pid = fork();
if(pid == 0)
printf("I am in child\n");
else
printf("I am in parent\n");
}
```

Answer: Both messages will be printed because pid is 0 for child and non zero for parent.

Q 13) Which message will show up on the screen?

```
main(){
int pid;
printf("Before fork()\n");
pid = fork();
if(pid == 0)
exec(name of program to execute);
else
printf("Only parent will get here");
}
```

Answer: Child will start executing program pointed by exec and will never reach `printf("Only parent will get here");` however parent will continue executing `printf("Only parent will get here");` means there will be only one copy of the message.

Q 14) Explain following program.

```
main(){
int pid;
pid = fork();
if(pid == 0){
printf("Child is sleeping\n");
sleep(5)       //do nothing for 5 sec
}
else{
printf("Parent is waiting\n");
waitpid(pid, NULL, 0)
printf("Back to the parent");
}
}
```

Answer: Parent stops at `waitpid(pid, NULL, 0)` for child to finish after which parent prints the message "Back to the parent".