Process Quizzes

Exercise

Which of the following pieces of information are NOT stored inside the Process Control Block (PCB)?

Choose JUST ONE option:

1. ☐ Process state
2. ☐ List of open files
3. ☐ CPU registers
4. ☐ Signal handlers
5. ☐ Process Identifier (PID)
6. ☐ They are all stored within the PCB
7. ☐ Administrative data, for instace related to CPU usage
8. ☐ Program Counter

Correct Answers:
6

Exercise

Suppose that a process becomes "zombie".
Indicate which of the following statements are correct. Note that wrong answers imply a penalty in the final score.

Choose one or more options:

1. ☐ The PCB of the process was deleted
2. ☐ The process performed a wait
3. ☐ The process has not a parent process
4. ☐ The process is terminated
5. ☐ The process before terminating had the "init" process as parent process
6. ☐ The process was inherited by the process “init”
7. ☐ Its PCB will be deleted only after its parent performs a wait or a waitpid

Correct Answers:
4, 7

Exercise

Suppose a process becomes an "orphan."
Please indicate which of the following statements are correct. Note that incorrect answers imply a penalty in the final score

Choose one or more options:

1. ☐ The process is waiting that the parent performs a wait
2. ☐ The process becomes an orphan because it did not perform a wait
3. ☐ The process will become "zombie" at its termination
4. ☐ The process is inherited by the "init" process
5. ☐ The process will not become "zombie" at its termination because the "init" process will inherit it

Correct Answers:
4, 5

Ex

Analyze the following piece of code.
Please indicate which of the following statements are correct. Note that incorrect answers imply a penalty in the final score.

```
if (fork() == 0) {
      /* first child */
      if (fork() == 0) {
            /* second child */
            ...
      } else {
            exit (1);
      }
}
wait ();
```

Choose one or more options:

1. ☐ The parent can suffer of deadlock
2. ☐ In case the first child has performed exit(1) , the second child will never become a zombie
3. ☐ To have a correct piece of code we must insert a second wait statement at the end
4. ☐ When the first child terminates, the second child is inherited by “init”
5. ☐ The parent waits the termination of the first child
6. ☐ The parent waits the termination of the first and the second child

Correct Answers:
2, 5

Exercise

Suppose you run the following segment of code:
Please indicate which of the following statements is correct.

```
if (fork()) {
      sleep (10);
      exit (1);
} else {
      exit (1);
}
```

Choose JUST ONE option:

1. ☐ The child process will become an orphan.
2. ☐ The parent process will become zombie.
3. ☐ The child process will become zombie when the parent terminates.
4. ☐ The parent process will become an orphan.
5. ☐ The child process will become zombie.

Correct Answers:
5

Exercise

If the following program is run, how many characters ‘P’ will be displayed on standard output?

```
int main () {
      int i;
      i=0;
      while (i<3 && fork()) {
            fork ();
            i++;
      }
      printf ("P");
      return 1;
}
```

Choose JUST ONE option:

1. ☐ 9
2. ☐ 16
3. ☐ 15
4. ☐ 8
5. ☐ 7

Correct Answers:
3

Exercise

Suppose a process does a waitpid. Please indicate which of the following statements are correct. Note that incorrect answers imply a penalty in the final score.

Choose one or more options:

1. ☐ The process will receive a SIGCHLD as soon as one of its children ends
2. ☐ The process can wait a maximum number of seconds
3. ☐ The process will exit from waitpid at the end of its first child
4. ☐ The process may get stuck on the waitpid even after a child has terminated

Correct Answers:
1, 4

Exercise

Describe what is an orphan process and then what is a zombie process.
Report two code segments: the first one generating an orphan process and the second one generating a zombie process.

Answer:

An orphan process is a child process in which the parent terminated before the child process. When this happens, orphan processes are "adopted" by a special os process (typically the init process). A zombie process is a child process that terminates before its parent called the system call wait().

```
// Orphan process
if(fork()){
      exit(0);
} else{
      // Orphan process
```

```
      sleep(1); // This orphan process will wait for a second so that we are sure
that the parent terminated
      exit(0);
}

// Zombie process
if(fork()){
      sleep(1); // Same as before but this time for the parent
      exit(0);
} else{
      //zombie process
      exit(0);
}
```

Exercise

Analyze the following segment of code. Indicate how many characters ‘X’ are displayed. Note that incorrect answers imply a penalty in the final score

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main () {
      int i;
      i = 0;
      setbuf(stdout,0);
      while (i<=2 && fork()) {
            if (fork ()) {
                  printf ("X");
            }
            i++;
      }
      return 1;
}
```

Choose JUST ONE option:

1. ☐ 14
2. ☐ 8
3. ☐ 6
4. ☐ 12
5. ☐ 7

Correct Answers:
5

Exercise

If you run the following program how many characters 'P' are displayed on standard output? Report a single integer value in your response.

```
int main () {
      int i;
      int pid;
      for (i=1; i<=2; i++) {
            pid = fork ();
            if (pid==0)
```

```
                fork();
        }
        printf("P");
        return 1;
}
```

Correct Answer:
9

Exercise

Indicate which of the following statements related to the fork() system call are correct. Note that incorrect answers imply a penalty in the final score.

Choose one or more options:

1. ☐ Parent and child share the code segment.
2. ☐ The fork() generates two new processes, to which are assigned two new pids.
3. ☐ Parent and child inherits the initial values of variables. (Meaning is: Parent and child inherit the values stored in the variables of the parent just before the fork())
4. ☐ The copy-on-write technique, after a fork(), permits to copy the initial values of variable in the parent and child processes.
5. ☐ Parent and child share the open file descriptors.

Correct Answer:
1, 3, 5

Exercise

Indicate which of the following statements related to process scheduling are correct. Note that incorrect answers imply a penalty in the final score.

Choose one or more options:

1. ☐ A process can move from the waiting state to the running state after an I/O operation or event completion.
2. ☐ The speedup that can be obtained from parallelization is limited by the fact that context switching operations reduce efficiency.
3. ☐ The process control block (PCB) contains all the information needed to manage the context switching, including a copy of the process stack and memory.
4. ☐ A process in the ready state does not use the CPU resource.
5. ☐ Different process queues are used to manage the access of the process to different resources when the process is in the waiting state.

Correct Answer:
2, 4, 5