|  |  |  |
| --- | --- | --- |
| **KING SAUD UNIVERSITY**  **COLLEGE OF COMPUTER AND INFORMATION SCIENCES Computer Science Department** | | |
| **CSC 227: Operating System** | **Tutorial# 4**  **Due: Sun, March 12 (12-1)** | **2nd Semester 1437-1438**  **Spring 2017** |

**Question#1:**

*Select (T) for true or (F) for False.*

|  |  |  |
| --- | --- | --- |
| True/False | | Statement |
| T | **F** | The Degree of Multiprogramming (DOM) designates the number of processes available in the ready and waiting queues. |
| T | **F** | Swapping processes preserves their locations in the memory address space |
| T | **F** | When a process is terminated and deallocates the memory it should swap out in the swap queue. |
| **T** | F | In Unix-based OSs, child processes spawned from a parent, may consume all parent’s resources. |
| T | **F** | If we define a variable x of type integer and initialize it with zero after calling CreateProcess() function then the value of x in the child process is zero. |
| **T** | F | The DOM stability depends on the difference between the number of allocated and deallocated processes. |
| T | **F** | The CPU resource utilization is directly proportional with the number of I/O-burst processes. |
| T | **F** | If all processes in the memory are in the ready state, then this leads to better overall system resource utilization. |
| T | **F** | If a process is interrupted, then medium term scheduler should swap it out. |
| **T** | F | In the absence of the function wait(), the child process and the parent process compete for the CPU. |

**Question#2:**

**Consider the following code**

#include <stdio.h>

#include <sys∕type.h>

void main(void) {

pid\_t pid; int i=0;

pid = fork();

if (pid==0)

pid = getpid();

else if (pid>0)

i=5;

else if (pid<0)

i=10;

return 0; }

**What are the values of i and pid:**

1. **If the fork() is successful and returned in the child process:**

**pid= process child ID (0)**

**i= 0**

1. **If the fork() is successful and returned in the parent process:**

**pid= Positive number ()**

**i= 5**

1. **If the fork() is unsuccessful:**

**pid= Negative number ()**

**i= 10**

**Question #3**

**In a multiprogramming environment, consider we have the following process P1:**

|  |  |  |
| --- | --- | --- |
| **#** | **Instruction** | **Execution time** |
| **1** | **Assignment** | **2 ms** |
| **2** | **Fork ()** | **3 ms** |
| **3** | **Wait (Null)** | **3 ms** |
| **4** | **Print** | **2ms** |

**Assume the following:**

* **fork () is executed correctly.**
* **Push time in any queue =2 ms.**
* **Wait time in printer queue =4 ms.**
* **Dispatch time = 1ms.**

**Complete the following table:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *Accumulative*  *Time* | *Process*  *State* | *Executing*  *Instruction* | *Queue*  *Name* | *Execution*  *Time* | *Dispatch Time* | *Push*  *Time* | *Wait*  *Time in Q* |
| ***0*** | ***Ready*** |  | ***Ready Q*** |  | ***1*** |  |  |
| *1* | *Running* | Assignment |  | *2ms* | *1ms* |  |  |
| *4* | *Running* | *Fork()* |  | *3ms* |  | *2ms* |  |
| *7* | *Waiting* |  | *Waiting Q* |  |  | *2ms* | *3ms* |
| *7* | *Ready* |  | *Ready Q* |  |  |  |  |
| *7* | *Running* | *Print* |  | *2ms* | *1ms* |  | *4ms* |
| *11* | *Waiting* |  | *Print Q* |  |  |  | *4ms* |
| *12* | *Terminated* |  |  |  |  |  |  |