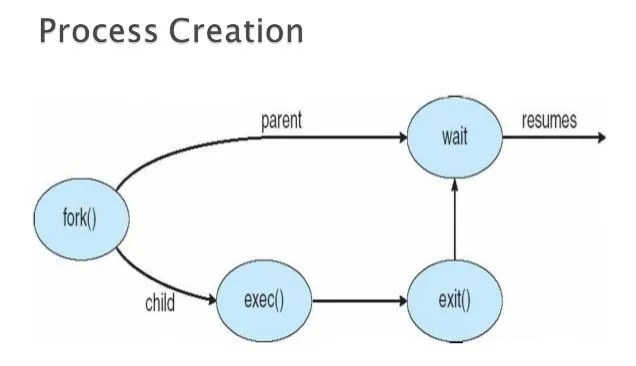
**PRE-LAB QUESTIONS**

1. What is process creation in operating systems?
   1. **A process may create several new processes, via a create-process system call, during the course of execution. The creating process is called a parent process, and the new processes are called the children of that process. The subsequent processes created forms a tree-like structure.**
2. Differentiate between Process, Parent Process and Child Process.
   1. **A process is an active program i.e. a program that is under execution. It is more than the program code as it includes the program counter, process stack, registers, program code, etc.**
   2. **All the processes in OS are created when a process executes the fork() system call except during the startup process. The process the uses the fork() system call is called the parent process. On the success of a fork() system call, the PID of the child process is returned to the parent process and 0 is returned to the child process else -1 is returned and the child process is not created.**
   3. **A child process is a process crated by a parent process in OS using a fork() system call. Also called as a subprocess or a subtask.**
3. List out some of the events that leads to process creation.



1. List out some of the causes of process termination.
   1. **Normal completion**
   2. **Unavailability of memory**
   3. **Exceed the execution time limit**
   4. **Violating memory access limits**
   5. **Protection error**
   6. **Arithmetic error**
   7. **I/O failure**
   8. **Misuse of data**
   9. **Exceeding the waiting time limit**
   10. **Invalid instruction execution**
   11. **Using a privileged instruction**
   12. **Interference by an OS or an operator**
   13. **Parent Process Termination**
   14. **Request from a parent process**
2. What is Process Suspension and Process Switching?
   1. **Whenever the processes in main memory are entered into the blocked state, the operating system suspends one process by putting it in the Suspended state and transferring it to disk. The free space present in the memory is used for bringing another process.**
   2. **It may occur any time when the operating system has gained control from the currently running process. Let us consider the system interrupts. There are two types of system interrupts which are as follows −**
      1. **Interrupt**
      2. **Trap**

**POST-LAB QUESTIONS**

1. Answer yes/no, and provide a brief explanation.
   1. Can two processes be concurrently executing the same program executable?
      1. **Yes, two processes can run the same program.**
   2. Can two running processes share the complete process image in physical memory (not just parts of it)?
      1. **No, in general but possible only when it is with copy-on-write during fork, and before any writes have been made.**
2. Consider a process executing on a CPU. Give an example scenario that can cause the process to undergo:
   1. A voluntary context switch.
      1. **A blocking system call.**
   2. An involuntary context switch.
      1. **Timer interrupt that causes the process to be switched out.**
3. Which of the following C library functions do not directly correspond to (similarly named) system calls? That is, the implementations of which of these C library functions are NOT straightforward invocations of the underlying system call?
   1. **system, which executes a bash shell command.**
   2. fork, which creates a new child process.
   3. exit, which terminates the current process.
   4. **strlen, which returns the length of a string.**
      1. **Reason: system and strlen are both C functions used to execute a bash shell command and to return the length of a string respectively. On the other hand, both fork and exit are system calls used to create a new child process and to terminate the current process respectively.**
4. List out the steps to be followed for process switch.
   1. **It is defined as that the processor switches from one thread/process to another thread or process. It makes the contents of the CPU registers and instruction pointer to be saved.** 
      1. **Save the content of the process that is currently running on the CPU. Update the PCB and other important fields.**
      2. **Move the PCB of the above process into the relevant queue such as the I/O queue, ready queue, etc.**
      3. **Select a new process for execution.**
      4. **Update the PCB of the selected process.**
      5. **Update the memory management data structures.**
      6. **Restore the content of the process that was previously running when it is loaded again on the processor. This is done by loading the previous values of the PCB and registers.**
5. How to know the return value from the unsuccessful termination of a child process through the wait() system call? Write a small program to demonstrate it.

|  |
| --- |
| **#define \_POSIX\_SOURCE**  **#include <sys/types.h>**  **#include <sys/wait.h>**  **#include <unistd.h>**  **#include <stdio.h>**  **#include <time.h>**  **main() {**  **pid\_t pid;**  **time\_t t;**  **int status;**  **if ((pid = fork()) < 0)**  **perror("fork() error");**  **else if (pid == 0) {**  **time(&t);**  **printf("child (pid %d) started at %s", (int) getpid(), ctime(&t));**  **sleep(5);**  **time(&t);**  **printf("child exiting at %s", ctime(&t));**  **exit(42);**  **}**  **else {**  **printf("parent has forked child with pid of %d\n", (int) pid);**  **time(&t);**  **printf("parent is starting wait at %s", ctime(&t));**  **if ((pid = wait(&status)) == -1)**  **perror("wait() error");**  **else {**  **time(&t);**  **printf("parent is done waiting at %s", ctime(&t));**  **printf("the pid of the process that ended was %d\n", (int) pid);**  **if (WIFEXITED(status))**  **printf("child exited with status of %d\n", WEXITSTATUS(status));**  **else if (WIFSIGNALED(status))**  **printf("child was terminated by signal %d\n",**  **WTERMSIG(status));**  **else if (WIFSTOPPED(status))**  **printf("child was stopped by signal %d\n", WSTOPSIG(status));**  **else puts("reason unknown for child termination");**  **}**  **}**  **}** |