CSCI 340
Lecturer: Simina Fluture

Lecture # 7

Read: Web Lecture
Class Notes
Topics: Process Operations

Process Operations

Activate: restart the process from the point at which it was suspended. The process is moved from the *suspended (swapped-out) in secondary memory* to the corresponding *active* state.

Swapped (suspended in secondary memory): it is often an operation performed by the OS to control the number of processes residing in the main memory, the degree of multiprogramming. Most of the time the suspension last only brief periods of time.
A process remains suspended until ANOTHER process activates it. Swapping is a high-priority operation.

Release: is required to move the process from either of the *blocked* state to the corresponding *ready* state.

Process Creation
A process can create new processes called the *children* of that process.

In terms of execution:
• the parent continues to execute concurrently with its children.
• the parent waits until some or all of its children have terminated.

In terms of resource utilization:
• the child may be able to obtain the resources directly from the OS.
• the child may be constrained to a subset of the resources of the parent process. (good for deadlock handling)

In terms of the address space:
• the child is a duplicate of the parent. (same address space & variables with same values)
• the child is a separate program (different address space)

UNIX
There are three important system calls used in the creation of a process:
fork( ), exec( ), wait( )
Process 1 is created as a part of the system boot-up. Every other process is created by P1 as a result of a fork system call. The *init* process manages all user login activity and create the initial user process for each user.
Fork Returned Value: **-1** cannot create a process
0 returned value to the child
1 – nnn The PID of the child returned to the parent process

Windows (NT/2000/XP)
in Win32API there is a **CreateProcess** function which in turn calls the Windows kernel functions ***NtCreateProcess*** and ***NTCreateThread***. There can be many different options for creating the process, so the CreateProcess function has many parameters, and some of these parameters can be quite complex. In contrast with the UNIX fork( ) call, where there are no parameters – in

windows the child's behavior is completely defined by parent's profile and default behavior.

MS-DOS
a system call exists to load a specified binary file into memory and execute it as a child process. The call suspends the parent until the child has finished execution.

Process Termination
When a process finishes executing, asks the OS to delete it by using the **exit** system call. The process may return data to its parent via the **wait** (provides the process id) system call.
All the resources of the process are deallocated.
A process can cause the termination of another process using the **abort** system call. Usually only the parent of the process that is to be terminated can invoke such system call.
On PC a user may quit its application. - Everything results in a system call to the OS to terminate the requesting process.

A parent may terminate the execution of one of its children because:
- the child exceeded the usage of some resources.
- the task assigned to the child is no longer required.
- the parent is exiting and the OS does not allow a child to continue if its parent terminates.

UNIX
A process may terminate by using the **exit** system call, and its parent process may wait for that event. If the parent terminates, all its children have assigned as their new parent the *init* process.
echo $? Displays the exit code of the last process.
Zombie State:

VMS
cascading termination.