System Calls for Process Management

Process Creation – The fork() System Call

- To Create a child process
- The process that invoked the system call – Parent Process
- Newly Created Process – Child Process

Pid = fork()

The fork() System Call

- Will not take any arguments
- Return value
  - Negative – Error in process Creation
  - Positive – Created successfully (to parent it is PID of child)
  - Zero - Child created (In child it is zero)

Kernel Operation – on fork()

- It allocates a slot in the process table for new process
- Assigns a unique ID for child process
- Makes a logical copy of the context of the parent process
- Returns ID number of newly created process to parent and 0 to the child.

Example

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main()
{
    /* fork a process */
    fork();
    /* the child and parent will execute every line of code after the fork (each separately)*/
    printf("Hello world!\n");
    return 0;
}
```

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main()
{
    /* fork a process */
    int pid;
   pid= fork();
    /* the child and parent will execute every line of code after the fork (each separately)*/
   if (pid==0)
   {
      printf("Hi I am Child - pid = %d \n", getpid());
      printf("Hi My parent id is  = %d\n ", getppid());
   }
   else if (pid>0)
   {
    printf("Parent Process - pid = %d\n", getpid());
    printf("Hello world!\n");
   }
```

Exec() System Call

- The exec system call is used to execute a file which is residing in an active process.
- When exec is called the previous executable file is replaced and new file is executed.
- Replaces the old file or program from the process with a new file or program. The entire content of the process is replaced with a new program.

Kernel Operation – on exec()

- Current process image is overwritten with a new process image.
- New process image is the one you passed as exec argument
- The currently running process is ended
- New process image has same process ID, same environment, and same file descriptor (because process is not replaced process image is replaced)
- The CPU stat and virtual memory is affected. Virtual memory mapping of the current process image is replaced by virtual memory of new process image.

Exec() System Call

- Takes the filename & path as the argument
- Return value - +ive integer – successful
- -ive – Error

Exit()- System Call

- Terminates the process normally
- Status is returned to parent program
  - 0 – Successful
  - Non Zero - Error

Exit()- System Call

- exit() performs following operations.

  * Flushes unwritten buffered data.

  * Closes all open files.

  * Removes temporary files.

  * Returns an integer exit status to the operating system.