Atma Ram Sanatan Dharma College

University of Delhi

Operating System

Practical File

Submitted By:

Jyotiswaroop Srivastav
College Roll No. 21/18023
Semester III
BSc. (Hons) Computer Science

Submitted To:

Ms. Parul Jain

1. Write a program (using fork() and/or exec() commands) where parent and child execute:

a) same program, same code.

b) same program, different code.

c) before terminating, the parent waits for the child to finish its task.

```c
/**
 * Write a program (using fork() and/or exec() commands)
 * where parent and child execute the same program, same
 * code.
 */

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>

int main()
{
    pid_t pidFork = fork();

    if (pidFork < 0)
        fprintf(stderr, "Error in fork()");
    else
        printf("Process ID: %d\n", getpid());

    return 0;
}
```

```
$ gcc -o main sameProgSameCode.c
$ ./main
Process ID: 150
Process ID: 151
```

```c
/**
 * Write a program (using fork() and/or exec() commands)
 * where parent and child execute the same program,
 * different code.
 */

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>

int main()
{
  pid_t pidFork = fork();

  if (pidFork < 0)
    fprintf(stderr, "Error in fork()\n");
  else if (pidFork > 0)
    printf("PARENT: Forked Child\n");
  else
  {
    printf("CHILD: Parent Process ID: %d\n", getppid());
    printf("CHILD: Process ID: %d\n", getpid());
    exit(0);
  }

  return 0;
}
```

```
$ gcc -o main sameProgDiffCode.c
$ ./main
PARENT: Forked Child
CHILD: Parent Process ID: 170
CHILD: Process ID: 171
```

```c
/**
 * Write a program (using fork() and/or exec() commands)
 * where before terminating, the parent waits for the
 * child to finish its task.
 */

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>
#include <sys/types.h>

int main()
{
    pid_t pidFork = fork();

    if (pidFork < 0)
        fprintf(stderr, "Error in fork()\n");
    else if (pidFork > 0)
    {
        wait(NULL);
        printf("PARENT: Child Exited\n");
    }
    else{
        printf("CHILD: Parent Process ID: %d\n", getppid());
        printf("CHILD: Process ID: %d\n", getpid());
        exit(0);
    }
    return 0;
}
```

```
$ gcc -o main waitForChild.c
$ ./main
CHILD: Parent Process ID: 191
CHILD: Process ID: 192
PARENT: Child Exited
```