Name: Ayush Sharma NYU ID Number: N15532582 Net ID: as21108

New York University
Tandon School of Engineering
Department of Computer Science and Engineering
Introduction to Operating Systems
Fall 2025
Assignment 3

Kernel Version: Linux 6.14.0-33-generic

```
ayushs2k1@ayushs2k1:~/Documents/lab3$ hostnamectl | grep Kernel
         Kernel: Linux 6.14.0-33-generic
ayushs2k1@ayushs2k1:~/Documents/lab3$
```

PART A

```
pid_t x=-11, y=-22, z=-33;
x = fork();
if(x>0) y=fork();
z = fork();
```

Step 1: First Fork

x = fork()

P0 (original):

- x = P1's PID (>0)

P1 (new child):

- x = 0

Both P0 and P1 continues

Name: Ayush Sharma NYU ID Number: N15532582 Net ID: as21108

Step 2: Second Fork

if (x > 0) y = fork()

For P0, x > 0 is TRUE, so it executes y = fork() and creates P2

P0 (original):

- y = P2's PID (>0)

P2 (new child):

- y = 0

P1:

- x = 0, so x > 0 is FALSE. It skips the fork and y remains -22

After step 2, we have processes P0, P1, and P2

Step 3: Third Fork

All three processes execute this fork.

P0 fork creates P3

P0:

- z = P3's PID (>0)

P3:

- z = 0, inherits P0's values: x > 0, y > 0

P1 fork creates P4

P1:

- z = P4's PID (>0)

P3:

- z = 0, inherits P1's values: x = 0, y = -22

P2 fork creates P5

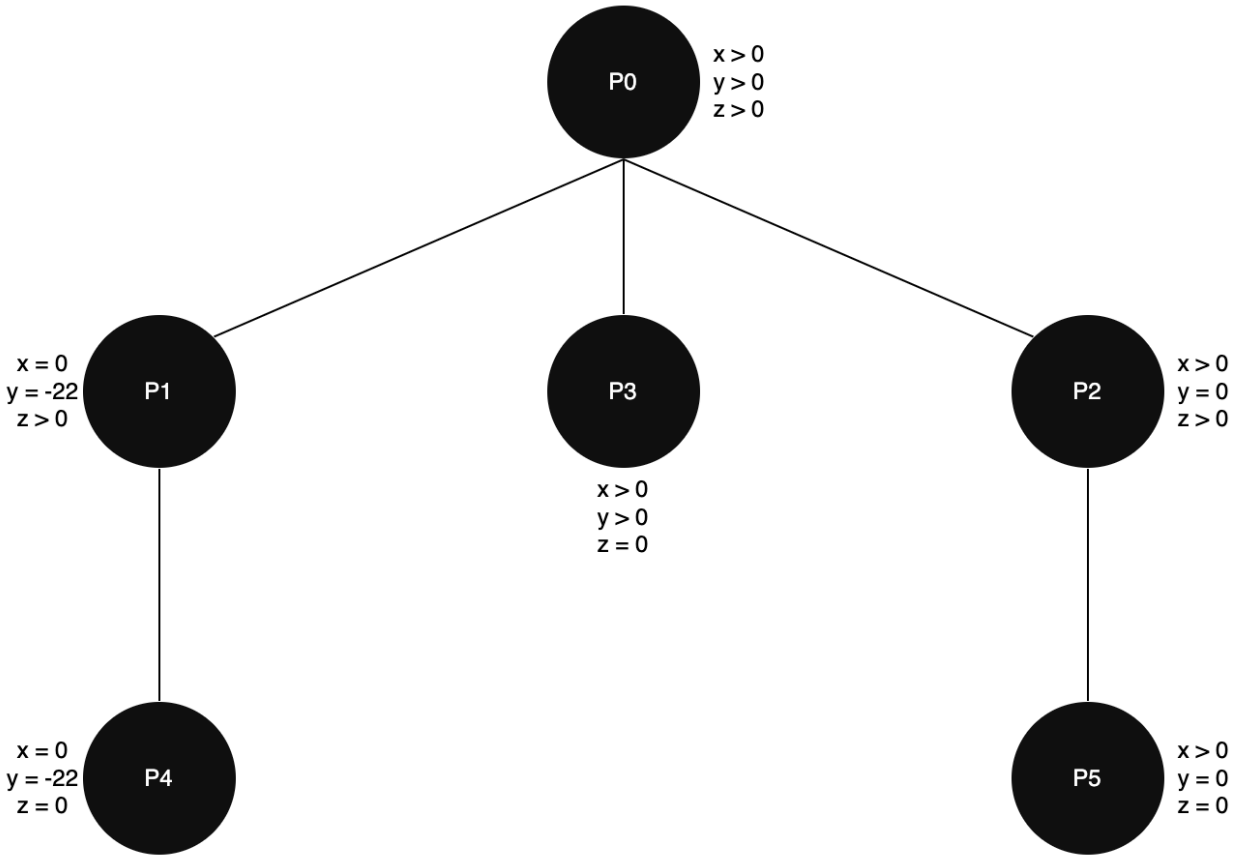
Name: Ayush Sharma NYU ID Number: N15532582 Net ID: as21108

P2:

- z = P5's PID (>0)

P5:

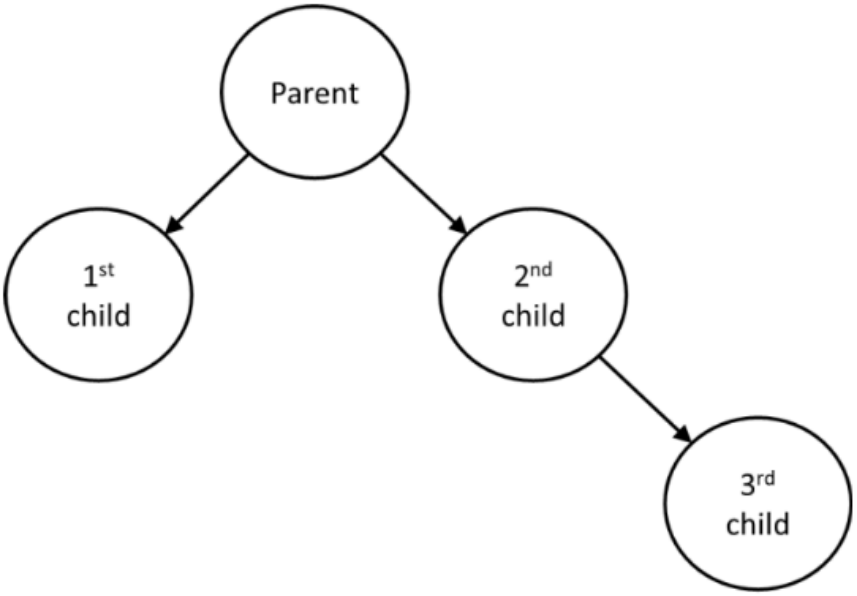
- z = 0, inherits P2's values: x > 0, y = 0



Name: Ayush Sharma NYU ID Number: N15532582 Net ID: as21108

PART B:

Write a subroutine that creates the process tree shown below and returns 0 for the parent, 1 for 1st child, 2 for the second child, and 3 for the third child.



Name: Ayush Sharma NYU ID Number: N15532582 Net ID: as21108

Code:

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

int create_process_tree(){
        pid_t pid1, pid2, pid3;

        // Parent creates the first child C1
        pid1 = fork();
        if (pid1 < 0){
                perror("fork failed");
                exit(1);
        }
        if (pid1 == 0){
                // This is first child C1
                return 1;
        }

        // Parent creates the second child C2
        pid2 = fork();
        if (pid2 < 0){
                perror("fork failed");
                exit(1);
        }
        if (pid2 == 0){
                // This is second child C2 and it creates child C3
                pid3 = fork();
                if (pid3 < 0){
                        perror("fork failed");
                        exit(1);
                }
                if (pid3 == 0){
                        // This is third child C3
                        return 3;
                }

                // C2 returns
                return 2;
        }

        // Parent process returns
        return 0;
}
```

```c
int main(){
        int result = create_process_tree();

        printf("Process PID: %d, Parent PID: %d, Role: ", getpid(), getppid());

        switch(result){
                case 0:
                        printf("Parent\n");
                        // Wait for both direct childern C1 and C2
                        wait(NULL);
                        wait(NULL);
                        break;
                case 1:
                        printf("First Child C1\n");
                        break;
                case 2:
                        printf("Second Child C2\n");
                        // Wait for the direct child C3
                        wait(NULL);
                        break;
                case 3:
                        printf("Third Child C3\n");
                        break;
        }

        return 0;
}
```

Name: Ayush Sharma NYU ID Number: N15532582 Net ID: as21108

Command used to compile the program: gcc -o lab4_b lab4_b.c
After compiling, it creates an executable file, lab4_b, as shown below.

```
ayushs2k1@ayushs2k1:~/Documents/lab4$ gcc -o lab4_b lab4_b.c
ayushs2k1@ayushs2k1:~/Documents/lab4$ ls -lrt
total 20
-rw-rw-r-- 1 ayushs2k1 ayushs2k1  1182 Oct  9 19:53 lab4_b.c
-rwxrwxr-x 1 ayushs2k1 ayushs2k1 70744 Oct  9 19:53 lab4_b
```

Running the executable:

```
ayushs2k1@ayushs2k1:~/Documents/lab4$ ./lab4_b
Process PID: 7772, Parent PID: 7510, Role: Parent
Process PID: 7773, Parent PID: 7772, Role: First Child C1
Process PID: 7774, Parent PID: 7772, Role: Second Child C2
Process PID: 7775, Parent PID: 7774, Role: Third Child C3
```

Name: Ayush Sharma NYU ID Number: N15532582 Net ID: as21108

PART C:

Write a program whose main routine obtains two parameters from the user, *n* and *d* (i.e., passed to your program when it was invoked from the shell, n>0), and creates a child process. The child process shall then create and print an arithmetic sequence of length *n* and whose elements are of type int, such that each element has a value of *kd*, where *k* is the element number (0 to *n*-1). For example, if *n*=5 and *d*=2, the sequence shall be 0,2,4,6, and 8. The parent waits for the child to exit and then prints 2 additional elements of the sequence, i.e., the total number of elements printed by the child and the parent is n+2. Do not use IPC in your solution to this problem (i.e., neither shared memory nor message passing).

Code:

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

int main(int argc, char *argv[]){
        int n, d;
        pid_t pid;
        int status;

        if(argc != 3){
                fprintf(stderr, "Usage: %s <n> <d>\n", argv[0]);
                fprintf(stderr, "n: length of sequence\n");
                fprintf(stderr, "d: common difference\n");
                exit(1);
        }

        n = atoi(argv[1]);
        d = atoi(argv[2]);

        if(n <= 0){
                fprintf(stderr, "n must be positive\n");
        }

        // Create child process
        pid = fork();

        if(pid < 0){
                perror("fork failed");
                exit(1);
        }

        if(pid == 0){
                //Child process: print n elements of the sequence
                for(int k=0; k<n; k++){
                        printf("%d, ", k*d);
                }
                exit(0);
        }
        else{
                // Parent process: print additional two elements of the sequence
                // Wait for the child to exit
                wait(&status);
                printf("%d, %d\n", n*d, (n+1)*d);
        }

        return 0;
}
```

Name: Ayush Sharma NYU ID Number: N15532582 Net ID: as21108

Command used to compile the program: gcc -o lab4_c lab4_c.c
After compiling, it creates an executable file, lab4_c, as shown below.

```
ayushs2k1@ayushs2k1:~/Documents/lab4$ gcc -o lab4_c lab4_c.c
ayushs2k1@ayushs2k1:~/Documents/lab4$ ls -lrt
total 40
-rw-rw-r-- 1 ayushs2k1 ayushs2k1  1182 Oct  9 19:53 lab4_b.c
-rwxrwxr-x 1 ayushs2k1 ayushs2k1 70744 Oct  9 19:53 lab4_b
-rw-rw-r-- 1 ayushs2k1 ayushs2k1   835 Oct  9 20:28 lab4_c.c
-rwxrwxr-x 1 ayushs2k1 ayushs2k1 70824 Oct  9 20:28 lab4_c
ayushs2k1@ayushs2k1:~/Documents/lab4$
```

Running the executable:

```
ayushs2k1@ayushs2k1:~/Documents/lab4$ ./lab4_c 5 2
0, 2, 4, 6, 8, 10, 12
ayushs2k1@ayushs2k1:~/Documents/lab4$
```

-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-**END**-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x