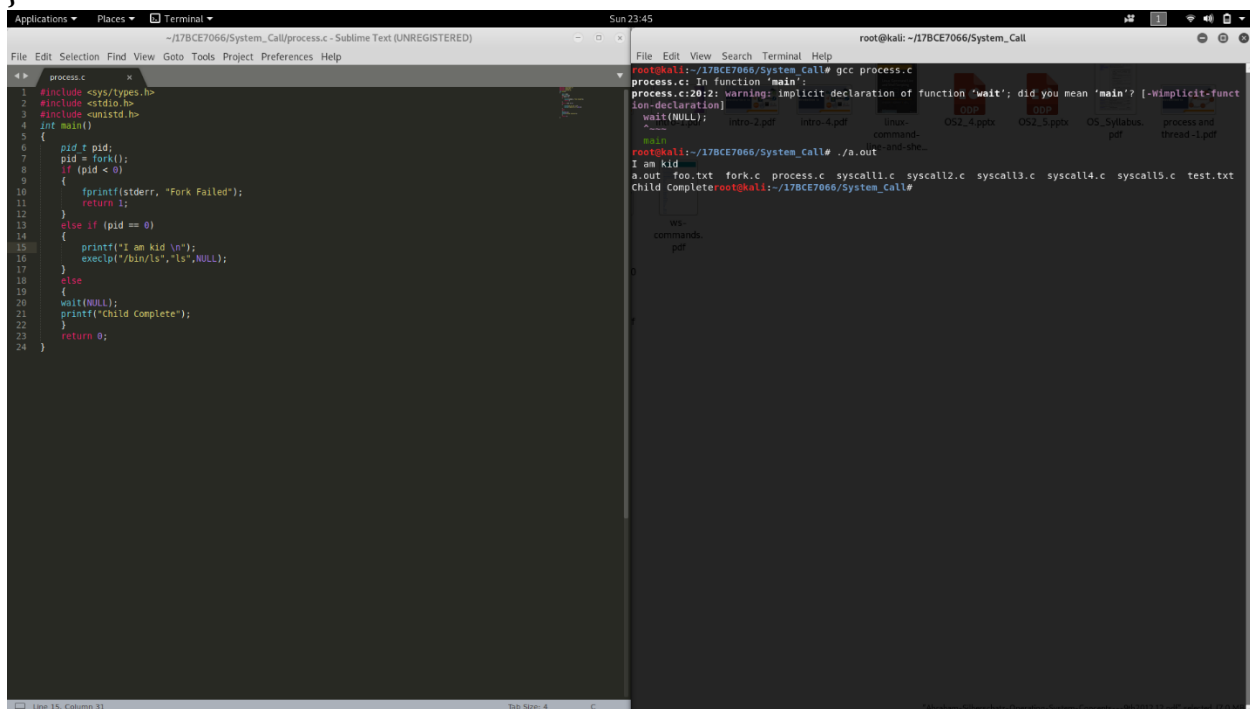


Inter-Process Communication (IPC)

Aditya Jain(17BCE7066)

Program 1: To create a child process and execute some task in it

```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>
int main()
{
    pid_t pid;
    pid = fork();
    if (pid < 0)
    {
        fprintf(stderr, "Fork Failed");
        return 1;
    }
    else if (pid == 0)
    {
        printf("I am kid \n");
        execlp("/bin/ls", "ls", NULL);
    }
    else
    {
        wait(NULL);
        printf("Child Complete");
    }
    return 0;
}
```



```
root@kali: ~/17BCE7066/System_Call# gcc process.c
process.c:20:2: warning: implicit declaration of function 'wait'; did you mean 'main'? [-Wimplicit-function-declaration]
    wait(NULL);
    ^~~~~
In file included from process.c:1:
/usr/include/unistd.h:100:1: note: 'wait' was declared here
    100 | wait(int);
        |
root@kali:~/17BCE7066/System_Call# ./a.out
I am kid
a.out foo.txt fork.c process.c syscall1.c syscall2.c syscall3.c syscall4.c syscall5.c test.txt
Child Complete
root@kali:~/17BCE7066/System_Call#
```

Program 2: To use shared-memory for IPC

Producer:

```
#include<stdio.h>
#include<stdlib.h>
#include <string.h>
#include<fcntl.h>
#include<sys/shm.h>
#include<sys/mman.h>
#include<sys/stat.h>

int main()
{
    const int SIZE = 4096;
    const char *name = "Adi";
    const char *message1 = "Hello \n";
    const char *message2 = "My name is Aditya Jain \n";
    int shm_fd;
    void *ptr;

    shm_fd = open(name, O_CREAT | O_RDWR, 0666);
    ftruncate(shm_fd, SIZE);
    ptr = mmap(NULL, SIZE, PROT_WRITE, MAP_SHARED, shm_fd, 0);
    sprintf(ptr, "%s", message1);
    ptr+=strlen(message1);
    sprintf(ptr,"%s",message2);
    ptr+=strlen(message2);
    return 0;
}
```

Consumer:

```
#include<stdio.h>
#include<stdlib.h>
#include <string.h>
#include<fcntl.h>
#include<sys/shm.h>
#include<sys/mman.h>
#include<sys/stat.h>

int main()
{
    const int SIZE = 4096;
    const char *name = "Adi";
    int shm_fd;
    void *ptr;

    shm_fd = open(name, O_CREAT | O_RDWR, 0666);
    ftruncate(shm_fd, SIZE);
    ptr = mmap(NULL, SIZE, PROT_READ, MAP_SHARED, shm_fd, 0);
```

```

printf( "%s", (char*)ptr);
unlink(name);
return 0;
}

```

The screenshot shows a Sublime Text editor with two files open: `shared_memory_producer.c` and `shared_memory_consumer.c`. The producer file includes `<stdio.h>`, `<stdlib.h>`, `<string.h>`, `<fcntl.h>`, `<sys/shm.h>`, `<sys/mman.h>`, and `<sys/stat.h>`. It defines a constant `SIZE = 4096`, a character array `name = "Adi"`, and two message strings. It opens a shared memory segment, truncates it to the specified size, maps it, and prints the first message. The consumer file includes `<sys/stat.h>` and defines the same `SIZE` and `name`. It opens the shared memory segment, maps it, prints the second message, and then unlinks the segment. A terminal window shows the compilation of both files with warnings about implicit declarations of `truncate`, `strncat`, and `unlink`. The execution output shows 'Hello' from the producer and 'My name is Aditya Jain' from the consumer.

```

shared_memory_producer.c
1 #include<stdio.h>
2 #include<stdlib.h>
3 #include<string.h>
4 #include<fcntl.h>
5 #include<sys/shm.h>
6 #include<sys/mman.h>
7 #include<sys/stat.h>
8
9 int main() {
10     const int SIZE = 4096;
11     const char *name = "Adi";
12     const char *message1 = "Hello \n";
13     const char *message2 = "My name is Aditya Jain \n";
14     int shm_fd;
15     void *ptr;
16
17     shm_fd = open(name, O_CREAT | O_RDWR, 0666);
18     ftruncate(shm_fd, SIZE);
19     ptr = mmap(NULL, SIZE, PROT_WRITE, MAP_SHARED, shm_fd, 0);
20     printf(ptr, "%s", message1);
21     ptr+=strlen(message1);
22     printf(ptr, "%s", message2);
23     ptr+=strlen(message2);
24     return 0;
25 }

```

```

shared_memory_consumer.c
7 #include<sys/stat.h>
8
9 int main() {
10     const int SIZE = 4096;
11     const char *name = "Adi";
12     int shm_fd;
13     void *ptr;
14
15     shm_fd = open(name, O_CREAT | O_RDWR, 0666);
16     ftruncate(shm_fd, SIZE);
17     ptr = mmap(NULL, SIZE, PROT_READ, MAP_SHARED, shm_fd, 0);
18     printf(ptr, "%s", (char*)ptr);
19     unlink(name);
20     return 0;
21 }

```

```

root@kali:~/17BCE7066/System_Call
root@kali:~/17BCE7066/System_Call# gcc -o producer shared_memory_producer.c
shared_memory_producer.c: In function 'main':
shared_memory_producer.c:18:2: warning: implicit declaration of function 'truncate'; did you mean 'strncat'? [-Wimplicit-function-declaration]
   truncate(shm_fd, SIZE);
   ^~~~~~
strncat
root@kali:~/17BCE7066/System_Call# gcc -o consumer shared_memory_consumer.c
shared_memory_consumer.c: In function 'main':
shared_memory_consumer.c:16:2: warning: implicit declaration of function 'truncate'; did you mean 'strncat'? [-Wimplicit-function-declaration]
   truncate(shm_fd, SIZE);
   ^~~~~~
strncat
shared_memory_consumer.c:19:2: warning: implicit declaration of function 'unlink'; did you mean 'munlock'? [-Wimplicit-function-declaration]
   unlink(name);
   ^~~~~~
munlock
root@kali:~/17BCE7066/System_Call# ./producer && ./consumer
Hello
My name is Aditya Jain
root@kali:~/17BCE7066/System_Call#

```

Program 3: To send a message using one pipe from parent to child and the child changes the case of every character and sends it back to the parent.

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

int main()
{
    int fd1[2];
    int fd2[2];

    char input_str[100];
    pid_t p;

    if (pipe(fd1)==-1)
    {
        fprintf(stderr, "Pipe Failed" );
        return 1;
    }
    if (pipe(fd2)==-1)
    {
        fprintf(stderr, "Pipe Failed" );
        return 1;
    }
    printf("Enter the message :");
    scanf("%s", input_str);
    p = fork();

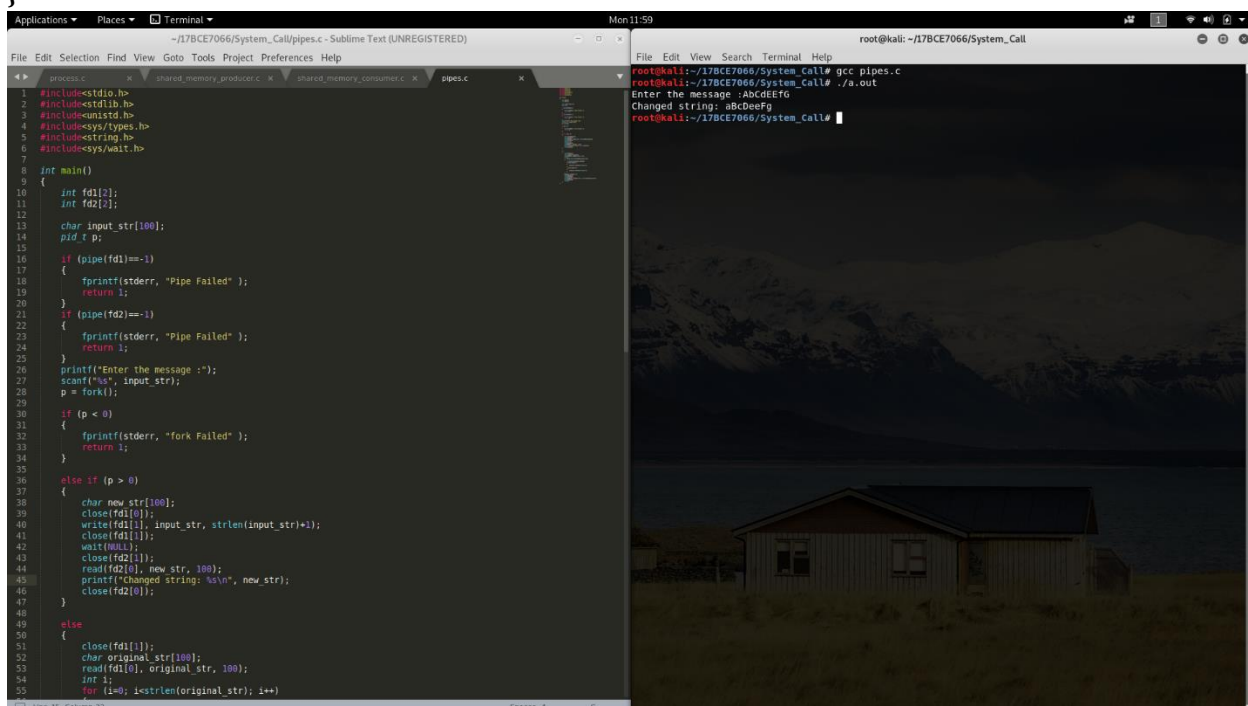
    if (p < 0)
    {
        fprintf(stderr, "fork Failed" );
        return 1;
    }

    else if (p > 0)
    {
        char new_str[100];
        close(fd1[0]);
        write(fd1[1], input_str, strlen(input_str)+1);
        close(fd1[1]);
        wait(NULL);
        close(fd2[1]);
        read(fd2[0], new_str, 100);
        printf("Changed string: %s\n", new_str);
    }
}
```

```

        close(fd2[0]);
    }
    else
    {
        close(fd1[1]);
        char original_str[100];
        read(fd1[0], original_str, 100);
        int i;
        for (i=0; i<strlen(original_str); i++)
        {
            int x=(int)(original_str[i]);
            if(x>=65&&x<=90)
            {
                original_str[i]=(char)(x+32);
            }
            if(x>=97&&x<=122)
            {
                original_str[i]=(char)(x-32);
            }
        }
        original_str[i]='\0';
        close(fd1[0]);
        close(fd2[0]);
        write(fd2[1], original_str, strlen(original_str)+1);
        close(fd2[1]);
        exit(0);
    }
}

```



```

root@kali: ~/J7BCE7066/System_Call
root@kali: ~/J7BCE7066/System_Call# gcc pipes.c
root@kali: ~/J7BCE7066/System_Call# ./a.out
Enter the message :aBcDeFg
Changed string: aBcDeFg
root@kali: ~/J7BCE7066/System_Call#

```