Lab – 04

Objectives:

1. Understanding the concept of Inter Process Communication (IPC)
2. Understanding the concept of Signals and their working
3. Understanding the concepts of Threads

Resources:

1. Video Lecture 09: https://youtu.be/q9lsekGzh0A?list=PL7B2bn3GwfBuJWtHADcXC44piWLRzr8
2. Video Lecture 10: https://youtu.be/lKSNjpGxt5o?list=PL7B2bn3GwfBuJWtHADcXC44piWLRzr8
3. Lecture Slides: https://www.arifbutt.me/wp-content/uploads/2024/08/Lecture-08-Thread-Management.pdf

Task 1:

I. What is the purpose of interprocess communication (IPC) in Unix systems?
II. Explain the difference between pipes and FIFOs (named pipes).
III. Draw a diagram to explain the working of pipes.

Task 2: Write a command which create a named pipe with a name "`fifo1`". Use this pipe to transfer some data from one process on one terminal to other process in any other terminal and display that data on screen.

Task 3: Create a file that contains the text "`This is, yes really! a text with?&* too many str$ange# characters ;-)`". Now write a set of command(s) that prints the above but do not print non-letters from above text. (Hint: use pipe)

Task 4: Write set of command(s) that receives a text file, and outputs all words on a separate line.

Task 5: Write down set of command(s) to make a sorted list of all files in `/etc` that contain the case insensitive string `conf` in their file name.

Task 6: Write down set of command(s) to put a sorted list of all bash users in bashusers.txt.

Task 7: What signals user can send to a process from keyboard give their name, number and their default action.

Task 8: What is the difference b/w "`$ kill pid`" and "`$ kill -15 pid`." What does "`$ kill -9 pid`" will do? (Note: pid could be any process id)

Task 9: See the man page and mention the different types of dispositions a signal can have by giving two examples signals for each.

Task 10: What are similarities and differences in a process and a thread? Give 2 use cases of muti-threading.

Task 11: What is difference between kernel and user level threads?

Task 12: Compile and execute the following code. Explain the output. And what are the advantages of compiling multithreaded programs using the –D_REENTRANT flag?

```
void* f1(void*);
void* f2(void*);
int main(){
pthread_t tid1, tid2;
pthread_create(&tid1, NULL, f1, NULL);
pthread_create(&tid2, NULL, f2, NULL);
pthread_join(tid1, NULL);
pthread_join(tid2, NULL);
printf("\nBye Bye from main thread\n");
return 0;
}
```

```
Void * f1(void * arg){
for(int i=0; i<5; i++){
printf("%s", "PUCIT");
fflush(stdout);
sleep(1);
}
pthread_exit(NULL);
}
void * f2(void * arg){
for(int i=0; i<5; i++){
printf("%s", "ARIF");
fflush(stdout);
sleep(1);
}
return NULL;
}
```

Task 13:

What will be the output of the following code snippet?

```
struct mystruct{
char character; int count;
};

void * f1(void * args){
struct mystruct p = *(struct mystruct*)args;
for (int i = 0; i < p.count; i++)
putc(p.character,stdout);
pthread_exit(NULL);
}
```

```
int main(){
pthread_t tid1, tid2;
struct mystruct t1_args, t2_args;
t1_args.character = 'X'; t1_args.count = 1000;
pthread_create(&tid1, NULL, f1, (void*)&t1_args);
t2_args.character = 'O'; t2_args.count = 800;
pthread_create(&tid2, NULL, f1, (void*)&t2_args);
pthread_join(tid1, NULL);
pthread_join(tid2, NULL);
printf("\nBye Bye from main thread.\n");
return 0;}
```