OS LAB Exp:3

```c
/*Create a new process and print its process id and its
parent's process id in C (use fork() and wait() system call).*/
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>
int main()
{
    pid_t pid;
    pid=fork();
    if(pid==0)
    {
        printf("child  id %d\n",getpid());
        printf("parent id %d\n",getppid());
    }
    else
    {
        wait(NULL);
        printf("child id %d\n",pid);
        printf("parent id %d\n",getpid());
    }
}
```

```
OUTPUT
child  id 2700
parent id 2699
child id 2700
parent id 2699
```

```c
/*Using C, create a new process and replace the child
process with the "ps" process. (use execl() system call).*/
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>
int main()
{
    pid_t pid;
    pid=fork();
    if(pid==0)
    {
        execl("/bin/ps","ps",NULL);
        printf("child  id %d\n",getpid());

    }
    else
    {
        wait(NULL);
        printf("parent id %d\n",getpid());
    }
}
```

```
OUTPUT
 PID TTY          TIME CMD
2766 pts/1    00:00:00 sh
2767 pts/1    00:00:00 sh
2769 pts/1    00:00:00 3b_replace
2770 pts/1    00:00:00 ps
parent id 2769
```

```c
/*Write a program to create a thread T1. The main process
passes two numbers to T1.T1 calculates the sum of these numbers
and returns the sum to the parent process for printing.*/
#include <stdio.h>
#include <pthread.h>
void *calculate(void *args) {
    int num1 = ((int*) args)[0];
    int num2 = ((int*) args)[1];
    int sum = num1 + num2;
    return (void*) sum;
}

int main() {
    int num1, num2;
    printf("Enter the first number: ");
    scanf("%d", &num1);
    printf("Enter the second number: ");
    scanf("%d", &num2);

    int args[2] = {num1, num2};
    void* sum;
    pthread_t t1;
    pthread_create(&t1, NULL, calculate, (void*) args);
    pthread_join(t1, &sum);

    printf("The sum of %d and %d is %d\n", num1, num2, (int) sum);

return 0;
}
```

```
OUTPUT
Enter the first number: 5
Enter the second number: 16
The sum of 5 and 16 is 21
```

```c
/*Write a program to create a thread T1. The main thread reads
an int number and checks whether it is prime or not. T1
calculates the factorial of the same number at the same time.*/
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
void *factorial(void *arg);
int main() {
    int num;
    printf("Enter a number: ");
    scanf("%d", &num);
    pthread_t tid;
    pthread_create(&tid, NULL, factorial, (void *)&num);
    int i, flag = 0;
    for(i = 2; i <= num/2; ++i) {
        if(num%i == 0) {
            flag = 1;
            break;
        }
    }
    if(num == 1) {
        printf("1 is not a prime number.\n");
    }
    else {
        if(flag == 0)
            printf("%d is a prime number.\n", num);
        else
            printf("%d is not a prime number.\n", num);
    }
    pthread_join(tid, NULL);
    return 0;
}
void *factorial(void *arg) {
    int num = *(int*)arg;
    int fact = 1, i;
    for(i = 1; i <= num; ++i) {
        fact *= i;
    }
    printf("Factorial of %d is %d.\n", num, fact);
    pthread_exit(NULL);
}
```

```
OUTPUT
Enter a number: 5
5 is a prime number.
Factorial of 5 is 120.
```

```c
/*IMPLEMENTATION OF OPENDIR AND READDIR SYSTEM CALLS*/
#include<sys/types.h>
#include<dirent.h>
#include<stdio.h>
int main(int c, char* arg[])
{
DIR *d;
struct dirent *r;
int i=0;
d=opendir(arg[1]);
printf("\n\t NAME OF ITEM \n");
while((r=readdir(d)) != NULL)
{
printf("\t %s \n",r->d_name);
i=i+1;
}
closedir(d);
printf("\n TOTAL NUMBER OF ITEM IN THE DIRECTORY IS %d \n",i);
return 0;
}
```

```
OUTPUT
NAME OF ITEM
.viminfo
thread2
.bashrc
linux-brprinter-installer-2.2.3-1.gz
...
...
TOTAL NUMBER OF ITEM IN THE THE DIRECTORY IS 177
```

```c
/*Write a program to create two Threads "Even" and "Odd". Thread "Even" calculates the sum of even numbers and Thread "Odd" calculates the sum of odd numbers.The main thread should display the even and odd numbers.*/
#include<pthread.h>
#include<stdio.h>
#define NUM_THREADS 3
int je,jo,evensum=0,sumn=0,oddsum=0,evenarr[50],oddarr[50];
void *Even(void *threadid)
{
int i,n;
je=0;
n=(int)threadid;
for(i=1;i<=n;i++)
{
if(i%2==0)
{
evenarr[je]=i;
evensum=evensum+i;
je++;
}
}
}
void *Odd(void *threadid)
{
int i,n;
jo=0;
n=(int)threadid;
for(i=0;i<=n;i++)
{
if(i%2!=0)
{
oddarr[jo]=i;
oddsum=oddsum+i;
jo++;
}
}
}
void *SumN(void *threadid)
{
int i,n;
n=(int)threadid;
for(i=1;i<=n;i++)
{
sumn=sumn+i;
}
}
int main()
{
pthread_t threads[NUM_THREADS];

int i,t;
printf("Enter a number\n");
scanf("%d",&t);
pthread_create(&threads[0],NULL,Even,(void *)t);
pthread_create(&threads[1],NULL,Odd,(void *)t);
pthread_create(&threads[2],NULL,SumN,(void *)t);
for(i=0;i<NUM_THREADS;i++)
{
pthread_join(threads[i],NULL);
}
printf("The sum of first N natural nos is %d\n",sumn);
printf("The sum of first N even natural nos is %d\n",evensum);
printf("The sum of first N odd natural nos is %d\n",oddsum);
printf("The first N even natural nos are----\n");
for(i=0;i<je;i++)
printf("%d\n",evenarr[i]);
printf("The first N odd natural nos are----\n");
for(i=0;i<jo;i++)
printf("%d\n",oddarr[i]);
pthread_exit(NULL);
}
```

OUTPUT

```
Enter a number
10
The sum of first N natural nos is 55
The sum of first N even natural nos is 30
The sum of first N odd natural nos is 25
The first N even natural nos are----
2
4
6
8
10
The first N odd natural nos are----
1
3
5
7
9
```

```c
/*Write a C Program to extract File information using stat() system call.*/
#include <sys/types.h>
#include <sys/stat.h>
#include <time.h>
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[])
{
    struct stat sb;
    if (argc != 2)
    {
        fprintf(stderr, "Usage: %s <pathname>\n", argv[0]);
        exit(EXIT_FAILURE);
    }
    if (stat(argv[1], &sb) == -1)
    {
        perror("stat");
        exit(EXIT_FAILURE);
    }
    printf("File type: ");
    switch (sb.st_mode & S_IFMT) {
    case S_IFBLK: printf("block device\n"); break;
    case S_IFCHR: printf("character device\n"); break;
    case S_IFDIR: printf("directory\n"); break;
    case S_IFIFO: printf("FIFO/pipe\n"); break;
    case S_IFLNK: printf("symlink\n"); break;
    case S_IFREG: printf("regular file\n"); break;
    case S_IFSOCK: printf("socket\n"); break;
    default: printf("unknown?\n"); break;
    }
    printf("I-node number: %ld\n", (long) sb.st_ino);
    printf("Mode: %lo (octal)\n",
    (unsigned long) sb.st_mode);
    printf("Link count: %ld\n", (long) sb.st_nlink);
    printf("Ownership: UID=%ld GID=%ld\n",
    (long) sb.st_uid, (long) sb.st_gid);
    printf("Preferred I/O block size: %ld bytes\n",
    (long) sb.st_blksize);
    printf("File size: %lld bytes\n",
    (long long) sb.st_size);
    printf("Blocks allocated: %lld\n",
    (long long) sb.st_blocks);
    printf("Last status change: %s", ctime(&sb.st_ctime));
    printf("Last file access: %s", ctime(&sb.st_atime));
    printf("Last file modification: %s", ctime(&sb.st_mtime));
    exit(EXIT_SUCCESS);
}
```

OUTPUT

```
File type:              regular file
I-node number:          4748721
Mode:                   100664 (octal)
Link count:             1
Ownership:              UID=1000   GID=1000
Preferred I/O block size: 4096 bytes
File size:              1223 bytes
Blocks allocated:       8
Last status change:     Sat Mar 18 07:45:00 2023
Last file access:       Sat Mar 18 07:45:00 2023
Last file modification: Sat Mar 18 07:45:00 2023
*
*       shaiju@Ubuntu:~$ ./a.out dir
File type:              directory
I-node number:          4719395
Mode:                   40775 (octal)
Link count:             2
Ownership:              UID=1000   GID=1000
Preferred I/O block size: 4096 bytes
File size:              4096 bytes
Blocks allocated:       8
Last status change:     Sun May  8 10:46:52 2022
Last file access:       Sat Mar 18 06:47:31 2023
Last file modification: Sun May  8 10:46:52 2022
```

```c
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include<fcntl.h>
#include <sys/types.h>
#include <sys/wait.h>
#include<stdlib.h>
int main(int argc, char *argv[])
{
    if(argc != 3)
    {
    printf("Argument Error.\n");
    exit(0);
    }
char *a = argv[1], *b = argv[2];
char buf[128] = "";
int rfd, wfd;
// only read
rfd = open(a, O_RDONLY);
// only write
wfd = open(b, O_WRONLY);
if(wfd == -1)
wfd = open(b, O_WRONLY | O_CREAT);
write(wfd, "START\n", 6);
while(read(rfd, buf, 1) > 0)
    {
    write(wfd, buf, strlen(buf));
    }
write(wfd, "\nSTOP\n", 6);
close(rfd);
close(wfd);
printf("Copied Contents from %s -> %s\n", a, b);
return 0;
}
```

OUTPUT

```
file1.txt:
Hello from OS Lab
by S4 CSE(A)
Bye!
* file2.txt
* START
Hello from OS Lab
by S4 CSE(A)
Bye!

STOP
```

```c
//Program to send a message from parent process to child process using pipe()
#include<stdio.h>
#include<unistd.h>
#include<sys/types.h>
#include<sys/wait.h>
int main()
{
    int fd[2],n;
    char buffer[100];
    pid_t p;
    pipe(fd); //creates a unidirectional pipe with two end fd[0] and fd[1]
    p=fork();
    if(p>0) //parent
    {
        printf("I am parent having id %d\n",getpid());
        printf("Parent Passing value to child\n");
        write(fd[1],"hello\n",6); //fd[1] is the write end of the pipe
        wait(NULL);
    }
    else // child
    {
    sleep(1);
    printf("I am child having id %d\n",getpid());
    printf("Child printing received value\n");
    n=read(fd[0],buffer,100); //fd[0] is the read end of the pipe
    write(1,buffer,n);
    }
}
```

OUTPUT

```
I am parent having id 29440
Parent Passing value to child
I am child having id 29441
Child printing received value
hello
```

```c
/*first process sends a number to second process
and calculates the factorial of that number.
(Use shared memoryconcept)*/
// Sender Process Program - ipcfactsend.c
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/shm.h>
int main()
{
key_t key = 12345;
printf("Writing to Shared Memory (Key = %d)\n",key);
int shmid = shmget(key, sizeof(int), 0666 |IPC_CREAT);
printf("shmid:\t%d\n", shmid);
void *shmad = shmat(shmid, NULL, 0);
printf("shmad:\t%p\n", shmad);
int n;
printf("\nEnter the value of N:");
scanf("%d", &n);
sprintf(shmad, "%d", n);
printf("\nWrite '%d' to SHM complete.\n\n", n);
return 0;
}
```

OUTPUT

```
snaiju@Ubuntu:~$ gcc ipcfactsend.c -o ipcfactsend
shaiju@Ubuntu:~$ ./ipcfactsend
Writing to Shared Memory (Key = 12345)
shmid: 131080
shmad:0x7f0901f55000

Enter the value of N:5

Write '5' to SHM complete.
```

```c
// Receiver Process Program - ipcfactreceive.c
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/shm.h>
int main()
{
key_t key = 12345;
printf("Reading from Shared Memory (Key = %d)\n",key);
int shmid = shmget(key, sizeof(int), 0666);
printf("shmid:\t%d\n", shmid);
if (shmid == -1)
{
printf("Error accessing shared memory Failure.\n");
exit(0);
}
void *shmad = shmat(shmid, NULL, 0);
printf("shmad:\t%p\n", shmad);
int n = atoi((char *)shmad);
printf("\nRead '%d' from SHM complete.\n", n);
shmdt(shmad);
shmctl(shmid, IPC_RMID, 0);
printf("SHM destroyed.\n\n");
long int res = 1;
for (int i = 1; i <= n; i++)
{
res *= i;
}
printf("fact(%d) = %ld\n\n", n, res);
return 0;
}
```

OUTPUT

```
snaiju@Ubuntu:~$ gcc ipcfactreveive.c -o ipcfactreceive
shaiju@Ubuntu:~$ ./ipcfactreceive
Reading from Shared Memory (Key = 12345)
shmid: 131080
shmad:0x7f0d1effe000

Read '5' from SHM complete.
SHM destroyed.

fact(5) = 120
```