5)Dining Philosopher Problem

```
#include<stdio.h>

#include<pthread.h>

#include<semaphore.h>

#include<unistd.h>

#include<stdlib.h>

#define N 5

#define THINKING 0

#define HUNGRY 1

#define EATING 2

#define LEFT (ph_num+4)%N #define RIGHT (ph_num+1)%N sem_t mutex,

phil_signal[N]; int state[N], phil[N]={0, 1, 2, 3, 4}; void test(int ph_num)

{

if(state[ph_num]==HUNGRY && state[LEFT]!=EATING && state[RIGHT]!=EATING)

{

state[ph_num]=EATING; sleep(2); printf("\nPhilosopher %d is eating\n", ph_num+1);

sem_post(&phil_signal[ph_num]);

}

}

void put_fork(int ph_num)

{

sem_wait(&mutex); state[ph_num]=THINKING; printf("\nPhilosopher %d has put the forks down.\n",

ph_num+1); test(LEFT); test(RIGHT); sem_post(&mutex);

}

void take_fork(int ph_num)
```

```
{
sem_wait(&mutex); state[ph_num]=HUNGRY; printf("\nPhilosopher %d is Hungry\n",
ph_num+1); test(ph_num); sem_post(&mutex); sem_wait(&phil_signal[ph_num]); sleep(1);
}
void * phils(void * pnum)
{
while(1)
{
int *ph_num=pnum;
sleep(1); take_fork(*ph_num); sleep(0);
put_fork(*ph_num);
}
}
int main()
{
sem_init(&mutex, 0, 1); int i=0; pthread_t phil_tid[N]; for(i=0; i<N; i++)
sem_init(&phil_signal[i], 0, 0); for(i=0; i<N; i++) pthread_create(&phil_tid[i], NULL, phils,
&phil[i]); for(i=0; i<N; i++) pthread_join(phil_tid[i], NULL); sem_destroy(&mutex); for(i=0;
i<N; i++) sem_destroy(&phil_signal[i]); return 0;
}
```

Output:

mml@mml-Vostro-3470:~$ gcc -o phil.out phil.c -lpthread

mml@mml-Vostro-3470:~$ ./phil.out Philosopher 1 is Hungry

Philosopher 1 is eating

Philosopher 2 is Hungry

Philosopher 3 is Hungry

Philosopher 3 is eating

Philosopher 4 is Hungry

Philosopher 5 is Hungry

Philosopher 1 has put the forks down.

Philosopher 5 is eating

Philosopher 3 has put the forks down. Philosopher 2 is eating

Philosopher 1 is Hungry Philosopher 5 has put the forks down.

Philosopher 4 is eating

Philosopher 3 is Hungry

Philosopher 2 has put the forks down.

Philosopher 1 is eating

Philosopher 5 is Hungry

6)IPC using Pipes:

```
#include<stdio.h>

#include<string.h>

#include<stdlib.h>

#include<unistd.h>

int main()

{

FILE *fp; int fd1[2], fd2[2], i=0;

char ch1, ch2, str1[100], str2[100], path[100]="/home/mml/pipe1/value.txt";

int ret1, ret2;

pid_t pid;

ret1=pipe(fd1);

ret2=pipe(fd2);

if(ret1==-1 || ret2==-1) printf("\nERROR\n"); pid=fork();

if(pid==0)

{

read(fd1[0], str2, 100);

fp=fopen(str2, "r"); while(!feof(fp))

{

ch2=fgetc(fp); write(fd2[1], &ch2, 1);

}

}

else

{

write(fd1[1], path, strlen(path)+1); while(read(fd2[0], &ch1, 1)>0) printf("%c", ch1);
```

```
}

}
```

Value.txt

```
JSPM
```

Output:

```
mml@mml-Vostro-3470:~$ cd pipe1

mml@mml-Vostro-3470:~/pipe1$ gcc -o os.out os.c

mml@mml-Vostro-3470:~/pipe1$ ./os.out JSPM
```

7)IPC using Shared memory using System V:

Server.c

```
#include<stdio.h>

#include<stdlib.h>

#include<sys/types.h>

#include<sys/ipc.h>

#include<sys/shm.h>

#include<unistd.h> #define MAXSIZE 27

void die(char *s) { perror(s); exit(1);

}

int main()

{

char c; int shmid; key_t key; char *shm, *s; key=5678; if((shmid=shmget(key,MAXSIZE,IPC_CREAT |

0666))<0) die("shmget"); if((shm=shmat(shmid,NULL,0))==(char *)-1) die("shmat"); s=shm;


for(c='a';c<='z';c++) *s++=c; while(*shm !='*')

sleep(1);

}
```

Output:

```
ml@mml-Vostro-3470:~$ cd ipc

mml@mml-Vostro-3470:~/ipc$ gcc -o server.out server.c

mml@mml-Vostro-3470:~/ipc$ ./server.out mml@mml-Vostro-3470:~/ipc$
```

Client.c

```
#include<stdio.h>

#include<stdlib.h>

#include<sys/types.h>

#include<sys/ipc.h>

#include<sys/shm.h>

#include<unistd.h> #define MAXSIZE 27

void die(char *s)

{ perror(s); exit(1);


}

int main()

{

int shmid; key_t key; char *shm, *s; key=5678;

if((shmid=shmget(key,MAXSIZE,0666))<0) die("shmget");

if((shm=shmat(shmid,NULL,0))==(char *)-1) die("shmat"); for(s=shm;*s!='\0';s++)

putchar(*s); putchar('\n');

*shm='*';

}
```

Output:

```
mml@mml-Vostro-3470:~/ipc$ gcc -o client.out client.c
mml@mml-Vostro-3470:~/ipc$./client.out
abcdefghijklmnopqrstuvwxyz
```