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
```

Reader Writer Problem:

```
#include<stdio.h>

#include<pthread.h>

#include<semaphore.h>

sem_t mutex,wrt;

int readcnt=0;

void *reader(void *data)

{

sem_wait(&mutex);

readcnt++;

if(readcnt==1)

sem_wait(&wrt);

sem_post(&mutex);

printf("reading\n"); sem_wait(&mutex);

readcnt--;

if(readcnt==0)

sem_post(&wrt);

sem_post(&mutex);

}

void *writer(void *data)

{

sem_wait(&wrt);
printf("Writer\n"); sem_post(&wrt);

}

int main()

{

sem_init(&wrt,0,1);

sem_init(&mutex,0,1);

pthread_t read[10],write[10]; int i=0;

for(i=0;i<10;i++)

{

pthread_create(&write[i],NULL,writer,NULL); pthread_create(&read[i],NULL,reader,NULL);

}

for(i=0;i<10;i++)

pthread_join(write[i],NULL);

for(i=0;i<10;i++)

pthread_join(read[i],NULL);

sem_destroy(&mutex);

sem_destroy(&wrt);

return 0;

}
```

Output:

```
mml@mml-Vostro-3470:~$ gcc -o RW.out RW.c -lpthread mml@mml-Vostro-3470:~$ ./RW.out
Writer
```

reading

reading

Writer

Writer

Writer

reading

reading

Writer

reading

Writer

reading

Writer

reading

Writer

reading

Writer

reading

Writer

reading

Producer Consumer Problem:

```
#include<stdio.h>

#include<pthread.h>

#include<semaphore.h>

sem_t mutex,wrt;

int readcount=0;

void *reader(void * data)

{

sem_wait(&mutex);

readcount++;

if(readcount==1)

sem_wait(&wrt);

sem_post(&mutex);

printf("\nReading.....\n"); sem_wait(&mutex);

readcount--;

if(readcount==0)

sem_post(&wrt);

sem_post(&mutex);

}

void * writer(void *data)
```

```
{

sem_wait(&wrt);

sem_post(&mutex);

}


int main()

{


sem_init(&wrt,0,1);

sem_init(&mutex,0,1);


pthread_t read[10],write[10]; int i=0;

for(i=0;i<10;i++)

{

pthread_create(&write[i],NULL,writer,NULL); pthread_create(&read[i],NULL,reader,NULL);

}

for(i=0;i<10;i++)

pthread_join(write[i],NULL);

for(i=0;i<10;i++)

pthread_join(read[i],NULL);

sem_destroy(&mutex);

sem_destroy(&wrt);
return 0;

}
```

Output:

```
mml@mml-Vostro-3470:~$ gcc -o pc.out pc.c -lpthread mml@mml-Vostro-3470:~$ ./pc.out
Reading.....

Reading.....

Reading.....

Reading.....

Reading.....

Reading.....

Reading.....

Reading.....

Reading.....
```

ipc

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

mml@mml-Vostro-3470:~/ipc$
```

create a folder name ipc

create two programmes named as server.c and client.c pipe1

OS.c

```
#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
```

____

create folder pipe1

create programmes os.c and value.txt Thread.c

```
#include<stdio.h>

#include<pthread.h>

#include<stdlib.h>

#include<unistd.h>

void *kidfunc(void *p)

{ printf("Kid Id is------->%d\n",getpid());

}

int main()

{ pthread_t kid; //datatype used to uniquely identify a thread
pthread_create(&kid,NULL,kidfunc,NULL);//if run successfully contains the id of a created
thread if fails no thread created

printf("Parent ID is------>%d\n",getppid());//process id of a calling process
pthread_join(kid,NULL);

printf("NO more kid!\n");

}

//pthread_create is used to create a thread

//pthread_join wait for termination of another thread
```

Output:

```
| ^~~~~~~

mml@mml-Vostro-3470:~$ gcc -o thread.out thread.c -lpthread mml@mml-Vostro-3470:~$
./thread.out Parent ID is------>6010

Kid Id is------->6082.
```

NO more kid!

Matrix1.c

```
#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <pthread.h>

#define SIZE 10

int A[SIZE][SIZE], B[SIZE][SIZE]; long int C[SIZE][SIZE];

void *mul_thread(void *arg) {

int i, row, col, *rcArgs;

long int return_val; //long int, since int cannot be type casted to void rcArgs = (int *) arg;

row = rcArgs[0];

col = rcArgs[1];

i

= rcArgs[2];

return_val = A[row][i] * B[i][col];

//return ((void *) return_val); pthread_exit((void *) return_val);

}

void accept_matrix(int M[SIZE][SIZE], int rows, int cols) {

int i, j;

printf("\n");

for(i=0;i<rows;i++) {
```

```
for(j=0;j<cols;j++) {

printf("Value at [%d][%d]: ",i+1,j+1); scanf("%d",&M[i][j]);

}

}

}

void display_matrix(int M[SIZE][SIZE], int rows, int cols) {

int i, j;

printf("\n");

for(i=0;i<rows;i++){

for(j=0;j<cols;j++){

printf("%2d ",M[i][j]);

}

printf("\n");

}

}

int main() {

int rows_A, cols_A, rows_B, cols_B; int rcArgs[3];

int i, j, k, *status;

pthread_t P[SIZE][SIZE][SIZE]; printf("\nEnter no. of rows in matrix A: "); scanf("%d",&rows_A);
printf("Enter no. of columns in matrix A: "); scanf("%d",&cols_A); accept_matrix(A, rows_A,
cols_A); printf("\nEnter no. of rows in matrix B: "); scanf("%d",&rows_B); printf("Enter no. of
columns in matrix B: "); scanf("%d",&cols_B); accept_matrix(B, rows_B, cols_B); if(cols_A ==
rows_B) {
```

```
for(i=0;i<rows_A;i++) {

for(j=0;j<cols_B;j++) {

for(k=0;k<cols_A;k++){

rcArgs[0] = i;

rcArgs[1] = j;

rcArgs[2] = k;

//Creating a new thread for every multiplication operation if(pthread_create(&P[i][j][k], NULL,
mul_thread, rcArgs) !=

0)

printf("\nCannot create thread.\n"); else

//Inserting delay

sleep(1);

}

}

}

} else {

printf("\n Matrix multiplication not possible."); exit(1);

}

printf("\nMatrix A:"); display_matrix(A, rows_A, cols_A); printf("\nMatrix B:"); display_matrix(B,
rows_B, cols_B); for(i=0;i<rows_A;i++) {

for(j=0;j<cols_B;j++) {

for(k=0;k<cols_A;k++){

//joining all the threads and retrieving values in status if(pthread_join(P[i][j][k],(void **) &status)
!= 0) perror("\nThread join failed.\n"); C[i][j] += (long int)status;

}

}

}

printf("\nResultant Matrix:\n"); for(i=0;i<rows_A;i++){

for(j=0;j<cols_B;j++){

printf("%2ld ",C[i][j]);

}

printf("\n");

}

exit(EXIT_SUCCESS);

}
```

Output:

```
mml@mml-Vostro-3470:~/OS$ gcc -o matrix1.out matrix1.c -lpthread
mml@mml-Vostro-3470:~/OS$ ./matrix1.out Enter no. of rows in matrix A: 2

Enter no. of columns in matrix A: 2

Value at [1][1]: 1

Value at [1][2]: 2

Value at [2][1]: 3

Value at [2][2]: 4

Enter no. of rows in matrix B: 2

Enter no. of columns in matrix B: 3
```

```
Value at [1][1]: 2

Value at [1][2]: 1

Value at [1][3]: 4

Value at [2][1]: 2

Value at [2][2]: 3

Value at [2][3]: 1

Matrix A:

1 2

3 4

Matrix B:

2 1 4

2 3 1

Resultant Matrix:

6 7 6

14 15 16
```

Fork.c

```
#include<sys/types.h>

#include<stdio.h>

#include<unistd.h>

#include<stdlib.h>

int main(){

int a[7];

for(int i=0;i<7;i++){

printf("Enter the intgers: %d",i); scanf("%d\n",&a[i]);

}

qsort(a,7,sizeof(int),compare); for(int i=0;i<7;i++){

printf("Sorted array is:"); printf("%d \n",a[i]);

}

pid_t pid;

pid=fork();

if(pid<0)

{

fprintf(stderr,"Fork Failed"); return 1;

}

else if(pid==0)

{

execlp("/bin/ls","ls",NULL); printf("child's pid : %d \n",getpid());
}

else

{

printf("child Process complete \n"); printf("parent's pid: %d \n",getppid());
execlp("ps","ps","-l",NULL); sleep(10);

}

return 0;

}
```

Output:

```
mml@mml-Vostro-3470:~$ gcc -o fork.out fork.c -lpthread mml@mml-Vostro-3470:~$ ./fork.out
child complete parent's pid 6723
```

Forks.c

```
#include<sys/types.h>

#include<stdio.h>

#include<unistd.h>

#include<stdlib.h>

int main()

{

pid_t pid;

pid=fork();

if(pid<0)

{ fprintf(stderr,"Fork Failed"); return 1;

}

else if(pid==0)

{

execlp("/bin/s","ls",NULL); printf("child's pid %d",getpid());

}

else
{
printf("child complete"); printf("parent's pid %d",getppid()); sleep(5);
}
return 0;
}
```

Output:

```
mml@mml-Vostro-3470:~$ gcc -o forks.out forks.c -lpthread mml@mml-Vostro-3470:~$
./forks.out child's pid 7076 child complete parent's pid 7056mml@mml-Vostro-3470:~$
```

binary11.c:

```
#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <string.h>

void bubbleSort(int arr[], int n) {

for (int i = 0; i < n - 1; i++) {

for (int j = 0; j < n - i - 1; j++) {

if (arr[j] > arr[j + 1]) {

int temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

}

}

int main() {

int n;

printf("Enter the number of elements in the array: "); scanf("%d", &n); int arr[n];
printf("Enter the elements of the array:\n"); for (int i = 0; i < n; i++) {

scanf("%d", &arr[i]);

}

int child_pid = fork();

if (child_pid == -1) {

perror("fork");

return 1;

}

if (child_pid == 0) {


char sorted_arr_str[1000] = ""; for (int i = 0; i < n; i++) {

char num_str[20];

sprintf(num_str, "%d ", arr[i]); strcat(sorted_arr_str, num_str);

}

} else {


wait(NULL);

bubbleSort(arr, n);

printf("Sorted array: "); for (int i = 0; i < n; i++) {

printf("%d ", arr[i]);

}

printf("\n");
}

return 0;

}
```

Output:

```
mml@mml-Vostro-3470:~$ gcc -o binary11.out binary11.c mml@mml-Vostro-3470:~$
./binary11.out (null)
```

Segmentation fault (core dumped) programmeos1.c

```
#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <string.h>

void bubbleSort(int arr[], int n) {

for (int i = 0; i < n - 1; i++) {

for (int j = 0; j < n - i - 1; j++) {

if (arr[j] > arr[j + 1]) {

int temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

}

}

int main() {

int n;

printf("Enter the number of elements in the array: "); scanf("%d", &n); int arr[n];

printf("Enter the elements of the array:\n"); for (int i = 0; i < n; i++) {
```

```
scanf("%d", &arr[i]);

}

int child_pid = fork();

if (child_pid == -1) {

perror("fork");

return 1;

}

if (child_pid == 0) {


char sorted_arr_str[1000] = ""; for (int i = 0; i < n; i++) {

char num_str[20];

sprintf(num_str, "%d ", arr[i]); strcat(sorted_arr_str, num_str);

}

} else {


wait(NULL);


bubbleSort(arr, n);

printf("Sorted array: "); for (int i = 0; i < n; i++) {

printf("%d ", arr[i]);

}
```

```
printf("\n");

}

return 0;

}
```

Output:

```
mml@mml-Vostro-3470:~/OS$ gcc -o programmeos1.out programmeos1.c
mml@mml-Vostro-3470:~/OS$ ./programmes1.out mml@mml-Vostro-3470:~/OS$
./programmeos1.out Enter the number of elements in the array: 5

Enter the elements of the array: 4

5

2

9

7

Sorted array: 2 4 5 7 9
```

zombie11.c

```
#include <sys/types.h>

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h> int compare(const void* num1, const void* num2){

int a = *(int*) num1;

int b = *(int*) num2;

if(a > b){

return 1;

} else if(a < b){

return -1;

} else {

return 0;

}

}

int main() {

int a[8],i;

printf("Enter integers : \n"); for(i=0; i<8; i++){

printf("Enter integer %d\n", i+1); scanf("%d", &a[i]);

}

qsort(a, 8, sizeof(int), compare); char str[128];
```

```
int j=0;

int index=0;

for(j=0; j<8; j++) {

index += sprintf(&str[index], "%d ", a[j]);

}

char *cmd = "./bina"; char *argu[3] = {"./bina", str, NULL}; pid_t pid;

pid=fork();

if(pid < 0) {

fprintf(stderr, "Failed Fork");

} else if(pid == 0) {

printf("Child Process : %d\n", getpid()); sleep(15);

} else if(pid > 0){

printf("Parent Process : %d\n", getppid()); execvp(cmd, argu);

sleep(15);

}

return 0;
```

Output:

```
mml@mml-Vostro-3470:~/OS$ gcc -o zombie11.out zombie11.c mml@mml-Vostro-3470:~/OS$
./zombie11.out Enter integers :

Enter integer 1

4

Enter integer 2
```

```
3

Enter integer 3

2

Enter integer 4

1

Enter integer 5

2

Enter integer 6

4

Enter integer 7

5

Enter integer 8

6

Parent Process : 7639

Child Process : 7656
```