UITI502 - Principles of Operating Systems.

Name: Dhamotharan B.

Reg No: 205002027.

class: IT - "A"

1) Write a C program in using unix systems calls and functions that will change permissions on file.

Code:

```
# include <stdio.h>.
# include <stdlib.h>.
int main () {
  char filename [16] = "file.txt";
   char cmd [32];
    int ret = 0;
    Printf (cmd, "chmod 666 %s", filename);
    ret = system (cmd);
    if (ret == 0)
    Printf (" permission of file changed succes \n");
    else
     Printf (" Unable to change permission \n");
     return 0;
    }
```

2)

```
#include <stdio.h>
# include <stdlib.h>
# include <pthread.h>
int sum=0 ;
int num;
int numcount;
int max= -100000;
int min= 100000;
int i ;
float avg;
void * avg-runner( void * avg)
{ int i ;
for( i= 0; i< numcount; i++){
scanf("%d", &num);
sum+= num ;
  avg= sum/sum-count ; }
  pthread_exit(0); }
void * avg min_runner( void * avg) {
  int i ;
  for( i = 0 ; i < numcount ; i++){
 printf("The program find the maximum, minimum,
average of a service of numbers.\n");
 printf("%d", & numcount);
 printf("Enter the number \n");
int avg = atoi(argv[1]);
int min= atoi(argv[2]);
int max = atoi(argv[3]);
Pthread_attr_t attr;
Pthread_attr_init (&attr);
Pthred_t thread1;
Pthread_t thread2;
Pthread_t thread3;
pthread_create(&thread1, &attr, avg-runner, &avg);
pthread_create(&thread2, &attr, min_runner, &min);
Pthred_join(thread1, Null);
Pthread_join(thread2, Null);
pthread_join(thread3, Null);
printf("The average is : %f\n", avg);
Printf("The minimum is : %d\n", min);
printf(" The maximum is : %d\n", max);
return 0; }.
```

3)

```
# include <stdio.b>
# includs < conio. b>
int state[N]
int phil[N]= {0,1,2,3,4}.
S ant mutex;
sem_t S[N];
void test (int pbum){
  if (state [Pbum] == HUNGRY && state[left]!=Eating &&
    state [Right] != Eating) {
                           {
              state [Phnum]= Eating;
               Sleep(2);
      Printf"(Philosopher %d take fork %d and %d\n",
                   Pbun+1, left+1, Phnum+1);
      Printf(" Philosopher %d is eating \n", phnum+1);
         sem-post (& S [Phnum]);  }
                                   }
void take-fork(int phnum)
{ sem_wait (& mutex);
   State[Phnum] = HUNGRY;
   printf(" philosophers %d is hungry \n", phnum+1);
   test (Phnum);
   sem-post (& mutex);
   sem_wait (& S[phnum]);
     Sleep(1); }
```

```
void   put-fork (int phnum);
     { sem - wait (& mutex);
        state [phnum] = THINKING;
        Printf("Philosopher %d putting fork %d, %d dow\n")
   Printf("philosopher %d is thinking \n", phnum+1);
     test (left);
        test (RIGHT);
         sem-post (& mutex); }

void* philosopher (void * num) {
            while (1) { int* i = num;
                  sleep (1);
                  take-fork(* i);
                   sleep(0);
                    put_fork (* i); } }

 int main() {
       int i ;
     pthread_t thread-id [N];
     sem-init [& mutex, 0, 1];
     for(i=0; i<N; i++);
          sem-init [& s[i],0,0];
      for(i=0 ; i<N ; i++){
         pthread-create(& thread-id[i], NULL, philosopher,
                                                 &phil[i];
          printf ("philosopher %d is thinking\n", i+1);
     for (i=0; i<N; i++)                        }
          pth_read_join (thread_id [i], NULL); }.
```

4)

a) Converting virtual address (in hexadecimal) to equivalent physical address.

Number of bits in logical address. =16 bits.

Page size = 4096 bytes. = 2^{12} bytes.

Logical address consists of page number, offset.

Number of bits used in offset = $\log_2$ (page size).

$= \log_2 2^{12}$ = 12 bits

Given, physical address i 0X E12C.

Binary = 1110 0001 0010 1100.

Page number is E(1110). offset is 12C(0001 0010 1100).

∴ physical address is 312C.

*) Given, virtual address is 0x3A9D.

Binary = 0011 1010 1001 1101.

Page number is 3(0011), offset A9D(1010 1001 1101)

Physical Address AA9D

*) virtual address is 0x A9D9.

Binary = 1010 1001 1101 1001

Page number is A(1010), offset is 9D9(1001 1101 1001)

Physical address = 59D9.

*) virtual address 0x 7001.

Binary = 0111 0000 0000 0001

Page number is 7(0111) offset 001(0000 0000 0001).

∴ physical address F001.

*) Virtual address is 0xACA1.

Binary = 1010 1100 1010 0001.

Page number is A(1010), offset CA1(1100 1010 0001)

Physical address 5CA1

b) 0x4ABC.

4th page is not availble in memory so any memory add rem starting with 4 will lead to page fault.

c) 3, A, 15, 5.

5)

Method	1	2	3	4	5	6	7	8	9	10	12	13	13	Total.
FCFS	2150	2069	1212	2296	2800	544	1618	356	1523	4965	3681			13011.
SSTF	2150	2069	2296	2800	3681	4965	1618	1523	1212	544	356			7586.
SCAN	2150	2296	2800	3681	4965	4999	2069	1618	1523	1212	544	356		7492.
C-SCAN	2150	2296	2800	3681	4965	4999	0	356	544	1212	1523	1618	2069	9917
LOOK.	2150	2296	2800	3681	4965	2069	1618	1523	1212	544	356			7424
C-LOOK.	2150	2296	2800	3681	4965	356	544	1212	1523	1618	2069			9137

In SCAN, C-SCAN, 4999 indicator that disk head has to moved to last track 4999. In C-SCAN disk head only scan in one-direction. As a result, after the disk header visited last track 4999, it has to be moved back to first track and scan in same direction.