**Operating Systems (CS306)**

**Assignment 1**

**Q1. Ans-**

**#include<stdio.h>**

**int main()**

**{**

**int n,bt[20],wt[20],tat[20],avwt=0,avtat=0,i,j;**

**printf("Enter total number of            processes(maximum 20):");**

**scanf("%d",&n);**

**printf("\nEnter Process Burst Time\n");**

**for(i=0;i<n;i++)**

**{**

**printf("P[%d]:",i+1);**

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

**}**

**wt[0]=0;    //waiting time for first process is 0**

**//calculating waiting time**

**for(i=1;i<n;i++)**

**{**

**wt[i]=0;**

**for(j=0;j<i;j++)**

**wt[i]+=bt[j];**

**}**

**printf("\nProcess\t\tBurst Time\tWaiting Time\tTurnaround Time");**

**//calculating turnaround time**

**for(i=0;i<n;i++)**

**{**

**tat[i]=bt[i]+wt[i];**

**avwt+=wt[i];**

**avtat+=tat[i];**

**printf("\nP[%d]\t\t%d\t\t%d\t\t%d",i+1,bt[i],wt[i],tat[i]);**

**}**

**avwt/=i;**

**avtat/=i;**

**printf("\n\nAverage Waiting Time:%d",avwt);**

**printf("\nAverage Turnaround Time:%d",avtat);**

**return 0;**

**}**

**Q.2. Ans:**

**#include <pthread.h>**

**#include <stdio.h>**

**int sum, sub, mul;**

**void \*sum(void \*param1);**

**void \*sub(void \*param2);**

**void \*mul(void \*param3);**

**/\* the thread \*/**

**int main(int argc, char \*argv[])**

**{**

**int i, array[3];**

**pthread\_t tid[3];  // thread identifier**

**if (argc != 3) {**

**fprintf(stderr,"usage: a.out <integer  value>\n");**

**return -1;**

**}**

**for(i = 1; i<argc ; i++)**

**array[i-1] = atoi(argv[i]);**

**pthread\_create(&tid[0],NULL,&sum,(void\*)array);**

**pthread\_create(&tid[1],NULL,&sub,(void\*)**

**array);**

**pthread\_create(&tid[2],NULL,&mul,(void\***

**)array);**

**/\* wait for the thread to exit \*/**

**for(i=0;i<3;i++)**

**pthread\_join (tid[i], NULL) ;**

**printf("sum = %d\n",sum);**

**printf("sub = %d\n",sub);**

**printf("mul = %d\n",mul);**

**}**

**/\* The thread will begin control in this function \*/**

**void \*sum(void \*param1)**

**{**

**int i;**

**int \*val = (int \*)param1;**

**sum = val[0]+val[1];**

**pthread\_exit (0) ;**

**}**

**void \*sub(void \*param1)**

**{**

**int i;**

**int \*val = (int \*)param1;**

**sub = val[0]-val[1];**

**pthread\_exit (0) ;**

**}**

**void \*mul(void \*param2)**

**{**

**int i;**

**int \*val = (int \*)param2;**

**mul = val[0]\*val[1];**

**pthread\_exit (0) ;**

**}**

**Q.3. Ans:**

**#include<stdio.h>**

**#include<signal.h>**

**#include<unistd.h>**

**void sighandler(int signum)**

**{**

**if (signum == SIGINT)**

**printf("In the sighandler: Received SIGINT\n");**

**}**

**int main(void)**

**{**

**signal(SIGINT, (void\*)sig\_handler);**

**while(1) {**

**printf(“Program running!”);**

**return 0;**

**}**

**}**

**Q.4. Ans: #include<stdio.h>**

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

**#include<unistd.h>**

**int main() {**

**pid\_t pid;**

**int n, first = 0, second = 1, next, c;**

**pid =fork();**

**if (pid < 0)**

**{**

**fprintf(stderr, "Fork Failed");**

**return 1;**

**} else if (pid == 0) {**

**for ( c = 0 ; c < n ; c++ )  
   {  
      if ( c <= 1 )  
        next = c;  
      else  
      {  
        next = first + second;  
        first = second;  
        second = next;  
     }  
     printf("%d\n",next);  
     }**

**}**

**else {**

**wait(NULL) ;**

**printf("Child Complete");**

**return 0;**

**}**

**}**

**Q.5. Ans: #include <pthread.h>**

**#include <stdio.h>**

**int fibo[100000];**

**void \*fib(void \*param);**

**/\* the thread \*/**

**int main(int argc, char \*argv[])**

**{**

**int i;**

**pthread\_t tid; /\* the thread identifier \*/**

**if (argc != 2) {**

**fprintf(stderr,"usage: a.out <integer value>\n");**

**return -1;**

**}**

**for(i=0;i<3;i++){**

**/\* create the thread \*/**

**pthread\_create(&tid,NULL,&fib,(void\*)argv[1]);**

**}**

**/\* wait for the thread to exit \*/**

**pthread\_join (tid, NULL) ;**

**for(i=0;i<argv[1];i++){**

**printf("%d\n",fibo[i]);**

**}**

**}**

**/\* The thread will begin control in this function \*/**

**void \*fib(void \*param1)**

**{**

**int i;**

**int val = atoi(param1);**

**fibo[0]=-1;**

**fibo[1]=1;**

**for(k=2;k<=val+1;k++){**

**fibo[k]=fibo[k-1]+fibo[k-2];**

**}**

**}**

**Q.6. Ans:**

**(1) A Web server that services each request in a separate thread.**

**(2) A parallelized application such as matrix multiplication where different parts of the matrix may be worked in parallel.**

**(3) An interactive GUI program such as a debugger where a thread is used to monitor user input, another thread represents the running application, and a third thread monitors performance.**

**Q.7. Ans:**

**struct usb\_ml {**

**/\* One structure for each connected device \*/**

**};**

**static struct usb\_device\_id ml\_table [] = {**

**{ USB\_DEVICE(ML\_VENDOR\_ID, ML\_PRODUCT\_ID) },**

**{ }**

**};**

**static int ml\_open(struct inode \*inode, struct file \*file)**

**{**

**/\* open syscall \*/**

**}**

**static int ml\_release(struct inode \*inode, struct file \*file)**

**{**

**/\* close syscall \*/**

**}**

**static ssize\_t ml\_write(struct file \*file, const char \_\_user \*user\_buf, size\_t**

**count, loff\_t \*ppos);**

**{**

**/\* write syscall \*/**

**}**

**static struct file\_operations ml\_fops = {**

**.owner =    THIS\_MODULE,**

**.write =    ml\_write,**

**.open =     ml\_open,**

**.release =  ml\_release,**

**};**

**static int ml\_probe(struct usb\_interface \*interface, const struct usb\_device\_id**

**\*id)**

**{**

**/\* called when a USB device is connected to the computer. \*/**

**}**

**static void ml\_disconnect(struct usb\_interface \*interface)**

**{**

**/\* called when unplugging a USB device. \*/**

**}**

**static struct usb\_driver ml\_driver = {**

**.name = "missile\_launcher",**

**.id\_table = ml\_table,**

**.probe = ml\_probe,**

**.disconnect = ml\_disconnect,**

**};**

**static int \_\_init usb\_ml\_init(void)**

**{**

**/\* called on module loading \*/**

**}**

**static void \_\_exit usb\_ml\_exit(void)**

**{**

**/\* called on module unloading \*/**

**}**

**module\_init(usb\_ml\_init);**

**module\_exit(usb\_ml\_exit);**