Operating System Lab-3

Thread

Patil Amit Gurusidhappa

19104004

B11

Q1

#include <sys/types.h>

#include <pthread.h>

#include <stdio.h>

#include <stdlib.h>

void \*th\_f(void \*arg) // this is fixed signature

{

    printf("process id %lu\n", getpid());

    printf("thread id %lu is executing \n", pthread\_self());

}

int main()

{

    pthread\_t tid;

    int ret;

    ret = pthread\_create(&tid, NULL, th\_f, NULL);

    if (ret != 0)

    {

        perror("thread 1 creation failed \n");

        exit(1);

    }

    ret = pthread\_create(&tid, NULL, th\_f, NULL);

    if (ret != 0)

    {

        perror("thread 1 creation failed \n");

        exit(1);

    }

    ret = pthread\_create(&tid, NULL, th\_f, NULL);

    if (ret != 0)

    {

        perror("thread 1 creation failed \n");

        exit(1);

    }

    ret = pthread\_create(&tid, NULL, th\_f, NULL);

    if (ret != 0)

    {

        perror("thread 1 creation failed \n");

        exit(1);

    }

    ret = pthread\_create(&tid, NULL, th\_f, NULL);

    if (ret != 0)

    {

        perror("thread 1 creation failed \n");

        exit(1);

    }

    pthread\_exit(NULL);

    return 0;

}

Q2

#include <sys/types.h>

#include <pthread.h>

#include <stdio.h>

#include <stdlib.h>

typedef struct value

{

    char name[10];

    int hall;

    int emp\_id;

    char branch[10];

} value;

void \*th\_f(void \*th) // this is fixed signature

{

    value \*v\_p;

    v\_p = (value \*)th;

    printf("process id %lu\n", getpid());

    printf("thread id %lu is executing \n", pthread\_self());

    printf(" Name -->%s \n", v\_p->name);

    printf(" Hall -->%d \n", v\_p->hall);

    printf(" Id -->%d \n", v\_p->emp\_id);

    printf(" Branch -->%s \n", v\_p->branch);

}

int main()

{

    pthread\_t tid;

    int ret;

    value \*v1, \*v2;

    v1 = (value \*)malloc(sizeof(value));

    v2 = (value \*)malloc(sizeof(value));

    strcpy(v1->name, "teacher1");

    v1->hall = 2;

    v1->emp\_id = 1;

    strcpy(v1->branch, "CSE");

    strcpy(v2->name, "teacher2");

    v2->hall = 2;

    v2->emp\_id = 2;

    strcpy(v2->branch, "ECE");

    ret = pthread\_create(&tid, NULL, th\_f, (void \*)v1);

    if (ret != 0)

    {

        perror("thread 1 creation failed \n");

        exit(1);

    }

    ret = pthread\_create(&tid, NULL, th\_f, (void \*)v2);

    if (ret != 0)

    {

        perror("thread 1 creation failed \n");

        exit(1);

    }

    pthread\_exit(NULL);

    return 0;

}

Q3

#include <stdio.h>

#include <fcntl.h>

#include <unistd.h>

#include <pthread.h>

#include <sys/types.h>

#include <semaphore.h>

#include <stdlib.h>

sem\_t posixsema;

int g\_amount = 1000;

void \*th\_deposit(void \*th)

{

    sem\_wait(&posixsema);

    g\_amount = g\_amount + 50 \* 50;

    sem\_post(&posixsema);

    return NULL;

}

void \*th\_withdraw(void \*th)

{

    sem\_wait(&posixsema);

    g\_amount = g\_amount - 20 \* 20;

    sem\_post(&posixsema);

    return NULL;

}

int main(void)

{

    pthread\_t tid1, tid2;

    int ret;

    /\* creating thread to deposit money \*/

    ret = sem\_init(&posixsema, 0, 1);

    if (-1 == ret)

    {

        printf("Failed to initialise Semaphore\n");

        exit(1); //Exit the Process

    }

    if (0 != pthread\_create(&tid1, NULL, th\_deposit, NULL))

    {

        perror("error in deposit thread");

        exit(1);

    }

    /\* creating thread to withdraw money \*/

    if (0 != pthread\_create(&tid2, NULL, th\_withdraw, NULL))

    {

        perror("error in deposit thread");

        exit(1);

    }

    pthread\_join(tid1, NULL);

    pthread\_join(tid2, NULL);

    printf("amount after transcation is : %d", g\_amount);

    return 0;

}

Q4

#include <stdio.h>

#include <fcntl.h>

#include <unistd.h>

#include <pthread.h>

#include <sys/types.h>

#include <semaphore.h>

#include <stdlib.h>

pthread\_mutex\_t mutex = PTHREAD\_MUTEX\_INITIALIZER;

int g\_amount = 1000;

void \*th\_deposit(void \*th)

{

    pthread\_mutex\_lock(&mutex);

    g\_amount = g\_amount + 50 \* 50;

    pthread\_mutex\_unlock(&mutex);

    return NULL;

}

void \*th\_withdraw(void \*th)

{

    pthread\_mutex\_lock(&mutex);

    g\_amount = g\_amount - 20 \* 20;

    pthread\_mutex\_unlock(&mutex);

    return NULL;

}

int main(void)

{

    pthread\_t tid1, tid2;

    int ret;

    /\* creating thread to deposit money \*/

    if (0 != pthread\_create(&tid1, NULL, th\_deposit, NULL))

    {

        perror("error in deposit thread");

        exit(1);

    }

    /\* creating thread to withdraw money \*/

    if (0 != pthread\_create(&tid2, NULL, th\_withdraw, NULL))

    {

        perror("error in deposit thread");

        exit(1);

    }

    pthread\_join(tid1, NULL);

    pthread\_join(tid2, NULL);

    printf("amount after transcation is : %d", g\_amount);

    return 0;

}