**Basic Knowledge**

strcpy(s1,s2) ==== s1 str e s2 copy kora

strcat(s1,s2) ==== s1 er sesh e s2 ke add kora

strcmp(s1,s2) ===== s1 s2 same hole 0, s1 <s2 hole less 0 , s1 >s2 hole bigger 0

system(“gedit tarunno/surzo.txt”)

system(pwd >> output.txt) eikhane append korar jonno (write korte hoile >)

gcc -o task3 task3.c -pthread -Wno-deprecated-declarations (to run semaphore)

Struct books

{

Char title[50]

Char aurthor[50]

Int value\_here

};

Struck Book b;

strcpy(b.title,”harry potter”);

b.value\_here= 2131;

**call by value**

Variable er value ta copy kore kaj kore so variable change hoyna

#include <stdio.h>

int sum(int a,int b); //just to introduce as it comes after main

int main(){

int x,y,z;

scanf("%d %d\n",&x,&y);

z=sum(x,y);

printf("%d\n",z);

return 0;

}

int sum(int a,int b){

int c;

c=a+b;

return c;

}

& var er sathe address return kore

\* var er sathe value return kore

**Pass by reference**

Jodi variable change korte chai thokon

#include <stdio.h>

void swap (int \*a,int \*b);

int main(){

int x,y,z;

scanf("%d\n",&x);

scanf("%d\n",&y);

swap(&x,&y);

printf("%d %d",x,y);

return 0;

}

void swap (int \*a,int \*b){

int temp;

temp=\*b;

\*b=\*a;

\*a=temp;

}

**Terminal related Codes**

//WRITE ON FILE GIVEN FROM TERMINAL

#include <stdio.h>

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

if (argc < 2) {

printf("Usage: %s <filename>\n", argv[0]);

return 1;

}

FILE \*file = fopen(argv[1], "w");

if (file == NULL) {

printf("Could not open file.\n");

return 1;

}

fprintf(file, "Written to file: %s\n", argv[1]);

fclose(file);

printf("Done writing to %s\n", argv[1]);

return 0;

}

**// Taking INTEGERS FROM TERMINAL AND MAKING ARRAY**

#include <stdio.h>

#include <stdlib.h>

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

if (argc != 6) {

printf("Usage: %s num1 num2 num3 num4 num5\n", argv[0]);

return 1;

}

int arr[5];

for (int i = 0; i < 5; i++) {

arr[i] = atoi(argv[i + 1]); // Convert string to int

}

printf("You entered:\n");

for (int i = 0; i < 5; i++) {

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

}

return 0;

}

**// NUMBER OF INTEGERS UNKNOWN FOR ABOVE PROBLEM**

#include <stdio.h>

#include <stdlib.h>

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

if (argc < 2) {

printf("Usage: %s num1 [num2 ... numN]\n", argv[0]);

return 1;

}

int count = argc - 1;

int arr[count];

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

arr[i] = atoi(argv[i + 1]); // Convert each argument to int

}

printf("You entered %d numbers:\n", count);

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

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

}

return 0;

}

**Input and output of file**

// ✅ Example 1: Write Int Array as Text

#include <stdio.h>

int main() {

int arr[] = {5, 10, 15, 20, 25};

int size = sizeof(arr) / sizeof(arr[0]);

FILE \*file = fopen("output.txt", "w");

if (file == NULL) {

printf("Failed to open file.\n");

return 1;

}

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

fprintf(file, "%d ", arr[i]); // Write each int followed by space

}

fclose(file);

return 0;

}

**//✅ Using fscanf() (Preferred for reading integers directly from file)**

#include <stdio.h>

int main() {

FILE \*file = fopen("numbers.txt", "r");

if (file == NULL) {

printf("Failed to open file.\n");

return 1;

}

int num;

while (fscanf(file, "%d", &num) == 1) { // Read integers one by one

printf("%d\n", num); // Print the integer

}

fclose(file);

return 0;

}

**//✅ Example 2: Using fgets() and atoi() (if integers are stored as strings)**

#include <stdio.h>

#include <stdlib.h>

int main() {

FILE \*file = fopen("numbers.txt", "r");

if (file == NULL) {

printf("Failed to open file.\n");

return 1;

}

char line[256];

while (fgets(line, sizeof(line), file)) {

int num = atoi(line); // Convert string to integer

printf("%d\n", num);

}

fclose(file);

return 0;

}

**Tasks solving**

**Finding GCD code**

#include <stdio.h>

int main()

{

int num1,num2;

char str[100];

int smaller;

int result;

printf("Enter two numbers: ");

scanf("%d, %d", &num1, &num2);

if(num1 < num2){

smaller = num1;

}

else{

smaller = num2;

}

for(int i = 1; i <= smaller; i++){

if(num1%i == 0 && num2%i == 0){

result = i;

}

}

printf("GCD is: %d\n", result);

return 0;

}

**Fibonacci recursive function**

#include <stdio.h>

int fibonacci(int n){

int res;

if(n == 0){

return 0;

}

else if(n == 1 || n == 2){

return 1;

}

else{

res = fibonacci(n-1) + fibonacci(n-2);

}

}

int main()

{

int result = fibonacci(8);

printf("%d", result);

return 0;

}

**Chips venting machine buy customer problem one at a time** **MUTEX**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <pthread.h>

#include <semaphore.h>

int t\_id[5]={1,2,3,4,5};

void \*customer\_buy(int \*id);

int chips = 5;

pthread\_mutex\_t mutex;

int main(){

pthread\_t customers[5];

pthread\_mutex\_init(&mutex,NULL);

for(int i=0;i<5;i++){

pthread\_create(&customers[i],NULL,(void \*)customer\_buy,&t\_id[i]);

}

for(int i=0;i<4;i++){

pthread\_join(customers[i],NULL);

}

pthread\_mutex\_destroy(&mutex);

return 0;

}

void \*customer\_buy(int \*id){

pthread\_mutex\_lock(&mutex);

if (chips>0){

chips--;

printf("Customer %d purchased Chips\n", \*id);

printf("Remaining stock: %d\n", chips);

}

pthread\_mutex\_unlock(&mutex);

}

**Factorial and prime number write in a file using fork call**

#include <sys/wait.h>

#include <stdbool.h>

#include <sys/types.h>

#include <unistd.h>

#include <stdlib.h>

#include <stdio.h>

void main()

{ FILE \*fp;

FILE \*fp2;

int a,b, status;

int user;

scanf("%d", &user);

a = fork();

char prime[256];

char factor[256];

if(a >0){

wait(&status);

b = fork();

if(b>0){

wait(&status);

fp = fopen("factorial.txt", "r");

fp2 = fopen("prime.txt", "r");

fgets(factor, sizeof(factor), fp);

fgets(prime, sizeof(prime), fp2);

printf("Factorial: %s", factor);

printf("Prime: %s", prime);

}

else if(b == 0){

fp2 = fopen("prime.txt", "w");

for(int i = 1; i <= user; i++){

bool prime = true;

for(int j = 2; j < i/2; j++){

if (i % j == 0){

prime = false;

}

}

if ( prime == true){

fprintf(fp2, "%d, ", i);

}

}

fclose(fp2);

}

}

else if(a==0){

int fac\_res = user;

for(int i = user-1; i > 0; i--){

fac\_res \*= i;

}

fp = fopen("factorial.txt", "w");

fprintf(fp, "%d\n", fac\_res);

fclose(fp);

}

}

**System call and thread to file read write**

#include <stdlib.h>

#include <stdio.h>

#include <unistd.h>

#include <pthread.h>

#include <semaphore.h>

sem\_t s;

void \* writeAddress(int \*id);

void \* writeID(int \*id);

FILE \*fp;

int main(){

int t\_id[2]={1,2};

system("mkdir moda");

system("touch moda/ghum.txt");

pthread\_t t1, t2;

sem\_init(&s,0,1);

pthread\_create(&t1,NULL,(void \*)writeID,&t\_id[0]);

pthread\_create(&t2,NULL,(void \*)writeAddress,&t\_id[1]);

pthread\_join(t1,NULL);

pthread\_join(t2,NULL);

sem\_destroy(&s);

return 0;

}

void \* writeID(int \*id){

sem\_wait(&s);

fp = fopen("moda/ghum.txt", "a");

fprintf(fp, "22101349");

fclose(fp);

sem\_post(&s);

}

void \* writeAddress(int \*id){

sem\_wait(&s);

system("pwd >> moda/ghum.txt");

sem\_post(&s);

}

**Car parking and car waiting using semaphore parking slot 1 and car waits for parking slot**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <pthread.h>

#include <semaphore.h>

int t\_id[5]={1,2,3,4,5};

void \*car\_park(int \*id);

int spot = 1;

sem\_t s;

int main(){

pthread\_t cars[5];

sem\_init(&s,0,1);

for(int i=0;i<5;i++){

pthread\_create(&cars[i],NULL,(void \*)car\_park,&t\_id[i]);

}

for(int i=0;i<4;i++){

pthread\_join(cars[i],NULL);

}

sem\_destroy(&s);

return 0;

}

void \*car\_park(int \*id){

if (sem\_trywait(&s) != 0) {

printf("Car %d is waiting....\n", \*id);

sem\_wait(&s);

}

printf("Car %d entered. Available spots: 0\n", \*id);

sleep(0.1);

printf("Car %d exited. Available spots: 1\n", \*id);

sem\_post(&s);

}

**Read write of a file using fork**

#include <sys/wait.h>

#include <stdbool.h>

#include <sys/types.h>

#include <unistd.h>

#include <stdlib.h>

#include <stdio.h>

void main()

{ FILE \*fp;

int a, status;

a = fork();

char buff[256];

if(a >0){

wait(&status);

printf("CSE321.txt opened for read access\n");

fp = fopen("a.txt", "r");

printf("CSE321.txt contains....\n");

while (fgets(buff, sizeof(buff), fp)) {

printf("%s\n", buff);

}

fclose(fp);

printf("CSE321.txt closed for read access\n");

}

else if(a==0){

printf("CSE321.txt opened for write access\n");

fp = fopen("a.txt", "w");

fprintf(fp, "aaaaa\nbbbbb\nccccc");

fclose(fp);

printf("CSE321.txt closed for write access\n");

}

}