**Fork, fibnocci**

#include<stdio.h>

#include<string.h>

#include<sys/types.h>

#include<stdlib.h>

#include<unistd.h>

#include<wait.h>

intmain(intargc,char\*argv[]){

printf("MainFunction:\n");

intretval=1;

pid\_tpid=fork();

if(pid<0){

printf("Errorinforkoperation\n");

}

if(pid==0){

printf("PIDforChildprocess:%d\nPIDofitsparentprocess:%d\n",getpid(),getppid());

execl("./binsearch",argv[1],NULL);

}

else{

printf("PIDofparentprocess:%d\n",getpid());

wait(&retval);

if(WIFEXITED(retval)==1)

{

printf("Childterminatednormally\n");

}

else{

printf("Childterminatedabnormally\n");

exit(0);

}

}

return0;

}

Certainly! Here's a simple C program to generate the Fibonacci series:

```c

#include <stdio.h>

// Function to generate Fibonacci series up to n terms

void fibonacci(int n) {

int first = 0, second = 1, next, i;

printf("Fibonacci Series up to %d terms:\n", n);

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

if (i <= 1)

next = i;

else {

next = first + second;

first = second;

second = next;

}

printf("%d ", next);

}

printf("\n");

}

int main() {

int n;

printf("Enter the number of terms for Fibonacci series: ");

scanf("%d", &n);

// Ensure n is a positive integer

if (n <= 0) {

printf("Please enter a positive integer.\n");

return 1;

}

fibonacci(n);

return 0;

}

```

This program takes an input from the user for the number of terms in the Fibonacci series and then generates the series up to that number of terms. Each term in the series is the sum of the two preceding terms.

Bs\_pthread

#include <stdio.h>

#include <pthread.h>

#define ARRAY\_SIZE 10

#define SEARCH\_ELEMENT 5

int array[ARRAY\_SIZE] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

int result = -1;

void \*left\_half\_search(void \*arg) {

for (int i = 0; i < ARRAY\_SIZE / 2; ++i) {

if (array[i] == SEARCH\_ELEMENT) {

result = i;

break;

}

}

pthread\_exit(NULL);

}

void \*right\_half\_search(void \*arg) {

for (int i = ARRAY\_SIZE / 2; i < ARRAY\_SIZE; ++i) {

if (array[i] == SEARCH\_ELEMENT) {

result = i;

break;

}

}

pthread\_exit(NULL);

}

int main() {

pthread\_t thread\_left, thread\_right;

if (pthread\_create(&thread\_left, NULL, left\_half\_search, NULL) != 0) {

perror("pthread\_create");

return 1;

}

if (pthread\_create(&thread\_right, NULL, right\_half\_search, NULL) != 0) {

perror("pthread\_create");

return 1;

}

pthread\_join(thread\_left, NULL);

pthread\_join(thread\_right, NULL);

if (result != -1) {

printf("Element %d found at index %d\n", SEARCH\_ELEMENT, result);

} else {

printf("Element %d not found in the array\n", SEARCH\_ELEMENT);

}

return 0;

}