**SYSTEM CALL: execv**

**FIRST FILE**

#include<stdio.h>

#include<math.h>

#include<stdlib.h>

#include<sys/types.h>

#include<string.h>

#include<unistd.h>

#include<sys/wait.h>

void quicksort(int a[20],int lb,int ub, int n)

{

int down,up,pivot,temp,k;

pivot=a[lb];

down=lb;

up=ub;

if(lb<ub)

{

while(down<up)

{

while(a[down]<=pivot)

down++;

while(a[up]>pivot)

up--;

if(down<up)

{

temp=a[down];

a[down]=a[up];

a[up]=temp;

}

}//end of outer while

a[lb]=a[up];

a[up]=pivot;

quicksort(a,lb,up-1,n);

quicksort(a,up+1,ub,n);

}//end of outer if

}//end of quicksort()

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

{

pid\_t pid;

int val[50], i, j;

char \*charArr[50];

char buffer[4];

printf("\n\*\*START\*\*\nArray: ");

for(i=1, j=0; i<argc; i++, j++)

{

val[j]=atoi(argv[i]);

}//conversion

for(i=0;i<argc-1;i++)

{

printf("%d ", val[i]);

}

printf("\n\n\*\*QUICK SORT\*\*\n");

quicksort(val,0,argc-2, argc-1);

printf("\nArray after Quick sorting: ");

for(i=0;i<argc-1;i++)

printf(" %d",val[i]);

for(i=0; i<argc-1; i++)

{

sprintf(buffer, "%d", val[i]);//3 blocks drawn in book.

charArr[i]=(char\*)malloc(sizeof(buffer));

strcpy(charArr[i], buffer);

}

charArr[i]=NULL; //as it is converted to string.

printf("\nArray after Conversion: ");

for(i=0; i<argc-1; i++)

printf("%s ", charArr[i]);

printf("\n");

pid=fork();

if(pid<0)

{

printf("Error in fork");

}

else

{

if(pid==0)

{

execv("new",charArr);

perror("execv"); //if this is executed error in program

printf("\n End Of Program \n");

}

else

{

sleep(10);

}

}

return 0;

}

**SECOND FILE**

#include<stdio.h>

#include<math.h>

#include<sys/types.h>

int binarySearch(int arr[], int l, int r, int x)

{

if (r >= l)

{

int mid = l + (r - l)/2;

// If the element is present at the middle itself

if (arr[mid] == x) return mid;

// If element is smaller than mid, then it can only be present in left subarray

if (arr[mid] > x) return binarySearch(arr, l, mid-1, x);

// Else the element can only be present in right subarray

return binarySearch(arr, mid+1, r, x);

}

// We reach here when element is not present in array

return -1;

}

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

{

int n, result;

int val[50], i, j;

for(i=0, j=0; i<argc; i++, j++)

{

val[j]=atoi(argv[i]);

}

printf("\n\*\*BINARY SEARCH\*\*\n\nEnter The Number to be Searched: ");

scanf("%d", &n);

result = binarySearch(val, 0, argc-1, n);

(result == -1)? printf("Oops!! Element is not present..\n\n")

: printf("Element Present!!\nIndex: %d\n\n", result);

return 0;

}

**OUTPUT**

[it@localhost ~]$ gcc -o new second.c

[it@localhost ~]$ gcc first.c

[it@localhost ~]$ ./a.out 1 7 5 3

\*\*START\*\*

Array: 1 7 5 3

\*\*QUICK SORT\*\*

Array after Quick sorting: 1 3 5 7

Array after Conversion: 1 3 5 7

\*\*BINARY SEARCH\*\*

Enter The Number to be Searched: 3

Element Present!!

Index: 1

[it@localhost ~]$ ./a.out 1 7 5 3

\*\*START\*\*

Array: 1 7 5 3

\*\*QUICK SORT\*\*

Array after Quick sorting: 1 3 5 7

Array after Conversion: 1 3 5 7

\*\*BINARY SEARCH\*\*

Enter The Number to be Searched: 4

Oops!! Element is not present..

[it@localhost ~]$