/\*

execve

2

sujay sanjay mahadik

code including execve

\*/

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

void quick(int array[], int firstIndex, int lastIndex)

{

int pivotIndex, temp, index1, index2;

if(firstIndex < lastIndex)

{

pivotIndex = firstIndex;

index1 = firstIndex;

index2 = lastIndex;

while(index1 < index2)

{

while(array[index1] <= array[pivotIndex] && index1 < lastIndex)

{

index1++;

}

while(array[index2]>array[pivotIndex])

{

index2--;

}

if(index1<index2)

{

temp = array[index1];

array[index1] = array[index2];

array[index2] = temp;

}

}

temp = array[pivotIndex];

array[pivotIndex] = array[index2];

array[index2] = temp;

quick(array, firstIndex, index2-1);

quick(array, index2+1, lastIndex);

}

}

int main()

{

pid\_t pid,p;

int status,i,j,n,arr[20];

char \*str[20];

printf("\nEnter the no of elements: ");

scanf("%d",&n);

printf("\nEnter the nos: ");

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

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

quick(arr,0,n-1); // Sorting

char \*ch;

asprintf(&ch, "%d", n);

str[1] = ch;

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

{

char c[sizeof(int)];

snprintf(c,sizeof(int),"%d",arr[i]);

str[i+2] = malloc(sizeof(c));

strcpy(str[i+2], c);

}

str[i+2]=NULL;

str[0]="b";

pid = fork();

if(pid == 0)

{

execve("/home/sujay/Desktop/sl2/print/code/b",str,NULL);

}

else

{

p = wait(&status);

printf("\n\nSorted elements are: \n");

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

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

printf("\t");

}

printf("\n");

return 0;

}

/\*

binary code for execve

\*/

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

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

{

int first, last, middle,array[20];

int i,k,search;

int n = atoi(argv[1]);

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

{

k = atoi(argv[i+2]);

array[i] = k;

}

printf("\nEnter the number you want to search: ");

scanf("%d",&search);

first = 0;

last = n - 1;

middle = (first+last)/2;

while (first <= last)

{

if (array[middle] < search)

first = middle + 1;

else if (array[middle] == search)

{

printf("Found at location: %d\n", middle+1);

break;

}

else

last = middle - 1;

middle = (first + last)/2;

}

if (first > last)

printf("Not found!\n");

}

