Experiment - 8

Operating System

Process Creation

Roehit Ranganathan
RA1911033010017
CSE SWE L2

1. Implement the C program in which the child process calculates the sum of odd numbers and the parent process calculate the sum of even numbers up to the number ‘n’.

```
roehit@LAPTOP-0SIPK43K:~/exp_8$ nano q1.c
roehit@LAPTOP-0SIPK43K:~/exp_8$ chmod 777 q1.c
roehit@LAPTOP-0SIPK43K:~/exp_8$ chmod +x q1.c
roehit@LAPTOP-0SIPK43K:~/exp_8$ gcc q1.c
roehit@LAPTOP-0SIPK43K:~/exp_8$ ./a.out
Enter the value of n: 8
Evensum: 20
Oddsum: 16
roehit@LAPTOP-0SIPK43K:~/exp_8$ cat q1.c
#include<stdio.h>
#include<unistd.h>

int main(){
int num;
printf("Enter the value of n: ");
scanf("%d",&num);
int no,i,e=0,o=0;
no=fork();

if(no>0){
for(i=1;i<=num;i++){
if(i%2==0)
e=e+i;
}
printf("Evensum: %d\n",e);
}

else{
for(i=1;i<=num;i++){
if(i%2!=0)
o=o+i;
}
printf("Oddsum: %d\n",o);
}
return 0;
}

roehit@LAPTOP-0SIPK43K:~/exp_8$
```

2. Implement the C program in which main program accepts the integers to be sorted Main program uses the fork system call to create a new process called a child process. Parent process sorts the integers using insertion sort and waits for child process using wait system call to sort the integers using selection sort.

```
roehit@LAPTOP-0SIPK43K: ~/exp_8
roehit@LAPTOP-0SIPK43K:~/exp_8$ nano q2.c
roehit@LAPTOP-0SIPK43K:~/exp_8$ chmod 777 q2.c
roehit@LAPTOP-0SIPK43K:~/exp_8$ chmod +x q2.c
roehit@LAPTOP-0SIPK43K:~/exp_8$ gcc q2.c
roehit@LAPTOP-0SIPK43K:~/exp_8$ ./a.out
Enter the number of Integers to Sort:::: 5
Enter number 1:9
Enter number 2:8
Enter number 3:5
Enter number 4:11
Enter number 5:3
Your Entered Integers for Sorting
9               8               5               11              3
Current Process ID is : 151
[ Forking Child Process ... ]
parent process 151 started

Parent of parent is 8
The Child Process

child process is 152
parent of child process is 151Child is sorting the list of Integers by Selection Sort::
The sorted List by Child::
3               5               8               9               11
Child Process Completed ...

parent of child process is 151The Parent Process
Parent 151 is sorting the list of Integers by Insertion Sort
The sorted List by Parent::
3               5               8               9               11
Parent Process Completed ...
roehit@LAPTOP-0SIPK43K:~/exp_8$ cat q2.c
#include<stdio.h>
#include <stdlib.h>
#include<sys/types.h>
#include<unistd.h>

int split(int[],int,int);
void insSort(int*,int,int);
void slcSort(int arr[],int low,int mid,int high){
int i,j,k,l,b[20];
l=low;
i=low;
j=mid+1;

while((l<=mid)&&(j<=high)){
if(arr[l]<=arr[j]){
b[i]=arr[l];
l++;
}
else{
b[i]=arr[j];
```

roehit@LAPTOP-0SIPK43K: ~/exp_8

```
b[i]=arr[l];
l++;
}
else{
b[i]=arr[j];
j++;
}
i++;
}

if(l>mid){
for(k=j;k<=high;k++){
b[i]=arr[k];
i++;
}
}
else{
for(k=l;k<=mid;k++){
b[i]=arr[k];
i++;
}
}

for(k=low;k<=high;k++){
arr[k]=b[k];
}
}

void seprt(int arr[],int low,int high){
int mid;
if(low<high){
double tmp;

mid=(low+high)/2;
seprt(arr,low,mid);
seprt(arr,mid+1,high);
slcSort(arr,low,mid,high);
}
}


void show(int a[],int size){
int i;
for(i=0;i<size;i++){
printf("%d\t\t",a[i]);
}
printf("\n");
}

int main(){
int pid, child_pid;
int size,i,status;
```

roehit@LAPTOP-0SIPK43K: ~/exp_8

```
printf("Parent of parent is %d\n",getppid());
sleep(30);
printf("The Parent Process\n");
printf("Parent %d is sorting the list of Integers by Insertion Sort\n",pid);

seprt(pArr,0,size-1);
printf("The sorted List by Parent::\n");
show(pArr,size);
printf("Parent Process Completed ...\n");
}

return 0;
}


int split(int a[ ],int lower,int upper ){
int i,p,q,t;

p=lower+1;
q=upper;
i=a[lower];

while(q>=p){
while(a[p]<i)
p++;

while(a[q]>i)
q--;

if(q>p){
t=a[p];
a[p]=a[q];
a[q]=t;
}
}

t=a[lower];
a[lower]=a[q];
a[q]=t;

return q;
}

void insSort(int a[],int lower,int upper){
int i;
if(upper>lower){
i=split(a,lower,upper);
insSort(a,lower,i-1);
insSort(a,i+1,upper);
}
}
roehit@LAPTOP-0SIPK43K:~/exp_8$ S
```

3. Implement the C program in which main program accepts an integer array. Main program uses the fork system call to create a new process called a child process. Parent process sorts an integer array and passes the sorted array to child process through the command line arguments of execve system call. The child process uses execve system call to load new program that uses this sorted array for performing the binary search to search the particular item in the array.

```
roehit@LAPTOP-0SIPK43K:~/exp_8$ nano q3parent1.c
roehit@LAPTOP-0SIPK43K:~/exp_8$ chmod 777 q3parent1.c
roehit@LAPTOP-0SIPK43K:~/exp_8$ chmod +x q3parent1.c
roehit@LAPTOP-0SIPK43K:~/exp_8$ cat q3parent1.c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>

int main(int argc, char *argv[]){
int val[10],ele;
pid_t pid;
char* cval[10];
char *newenviron[] = { NULL };
int i,j,n,temp;
printf("\nEnter the size for an array: ");
scanf("%d",&n);
printf("\nEnter %d elements : ", n);
for(i=0;i<n;i++)
scanf("%d",&val[i]);

printf("\nEntered elements are: ");
for(i=0;i<n;i++)
printf("\t%d",val[i]);

for(i=1;i<n;i++){
for(j=0;j<n-1;j++){
if(val[j]>val[j+1]){
temp=val[j];
val[j]=val[j+1];
val[j+1]=temp;
}
}
}
printf("\nSorted elements are: ");
for(i=0;i<n;i++)
printf("\t%d",val[i]);
printf("\nEnter element to search: ");
scanf("%d",&ele);
val[i] = ele;
for (i=0; i < n+1; i++){
char a[sizeof(int)];
snprintf(a, sizeof(int), "%d", val[i]);
cval[i] = malloc(sizeof(a));
strcpy(cval[i], a);
}
cval[i]=NULL;

pid=fork();
if(pid==0){
execve(argv[1], cval, newenviron);
perror("Error in execve call...");
```

```
perror("Error in execve call...");
}
}
roehit@LAPTOP-0SIPK43K:~/exp_8$ nano q3child1.c
roehit@LAPTOP-0SIPK43K:~/exp_8$ chmod 777 q3child1.c
roehit@LAPTOP-0SIPK43K:~/exp_8$ chmod +x q3child1.c
roehit@LAPTOP-0SIPK43K:~/exp_8$ cat q3child1.c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(int argc, char *argv[],char *en[]){
int i,j,c,ele;
int arr[argc];

for(j = 0;j<argc-1;j++){
int n=atoi(argv[j]);
arr[j]=n;
}
ele=atoi(argv[j]);
i=0;
j=argc-1;
c=(i+j)/2;
while(arr[c]!=ele && i<=j){
if(ele > arr[c])
i = c+1;
else
j = c-1;
c = (i+j)/2;
}
if(i<=j)
printf("\nElement Found in the given Array...!!!\n");
else
printf("\nElement Not Found in the given Array...!!!\n");
}


roehit@LAPTOP-0SIPK43K:~/exp_8$ gcc q3parent1.c
roehit@LAPTOP-0SIPK43K:~/exp_8$ gcc -o new q3child1.c
roehit@LAPTOP-0SIPK43K:~/exp_8$ ./a.out new

Enter the size for an array: 5

Enter 5 elements : 5
4
3
2
1

Entered elements are:   5       4       3       2       1
Sorted elements are:    1       2       3       4       5
Enter element to search: 4
roehit@LAPTOP-0SIPK43K:~/exp_8$
Element Found in the given Array...!!!
```

4. Write a program to print the Child process ID and Parent process ID in both Child and Parent processes..

```
roehit@LAPTOP-0SIPK43K: ~/exp_8
roehit@LAPTOP-0SIPK43K:~/exp_8$ nano q4.c
roehit@LAPTOP-0SIPK43K:~/exp_8$ chmod 777 q4.c
roehit@LAPTOP-0SIPK43K:~/exp_8$ chmod +x q4.c
roehit@LAPTOP-0SIPK43K:~/exp_8$ gcc q4.c
roehit@LAPTOP-0SIPK43K:~/exp_8$ ./a.out
hello bedore fork
i : 32556


parent has started
Child has started
getpid : 187 getppid : 8
child printing first time

getpid : 188 getppid : 187
Hi after fork i : 188
roehit@LAPTOP-0SIPK43K:~/exp_8$
child printing second time
getpid : 188 getppid : 1
Hi after fork i : 0
^C
roehit@LAPTOP-0SIPK43K:~/exp_8$ cat q4.c
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>

int main(){
int procss;
printf("hello bedore fork\n");
printf("i : %d\n",procss);

procss=fork();
printf("\n");

if(procss==0){
printf("Child has started\n");
printf("child printing first time \n");

printf("getpid : %d getppid : %d \n",getpid(),getppid());
sleep(5);
printf("\nchild printing second time\n");
printf("getpid : %d getppid : %d \n",getpid(),getppid());
}

else{
printf("parent has started\n");
printf("getpid : %d getppid : %d \n",getpid(),getppid());
printf("\n");
}
printf("Hi after fork i : %d\n",procss);
return 0;
}
```