**Assignment 2-A**

#include<stdio.h>

#include<sys/types.h>

#include<unistd.h>

#include<stdlib.h>

void bass(int arr[30],int n)

{

int i,j,temp;

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

{

for(j=0;j<n-1;j++)

{

if(arr[j]>arr[j+1])

{

temp=arr[j];

arr[j]=arr[j+1];

arr[j+1]=temp;

}

}

}

printf("\n Ascending Soring \n");

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

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

}

void bdsc(int arr[30],int n)

{

int i,j,temp;

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

{

for(j=0;j<n-1;j++)

{

if(arr[j]<arr[j+1])

{

temp=arr[j];

arr[j]=arr[j+1];

arr[j+1]=temp;

}

}

}

printf("\n Descending Sorting \n");

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

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

printf("\n\n");

}

void forkeg()

{

int arr[25],arr1[25],n,i,status;

printf("\nEnter Size : - \n");

scanf("%d",&n);

printf("\nEnter Array : - \n");

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

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

int pid=fork();

if(pid==0)

{

printf("\nChild Id = %d\n",getpid());

bdsc(arr,n);

printf("\n Bubble Sorting \n");

printf("\n");

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

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

printf("\n");

}

else

{

sleep(10);

printf("Parent Id = %d\n",getppid());

bass(arr,n);

printf("\n Bubble Sort : \n");

printf("\n");

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

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

printf("\b");

system("ps -x");

}

}

int main()

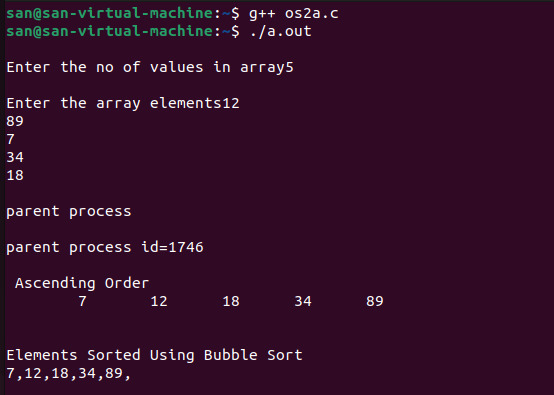
{

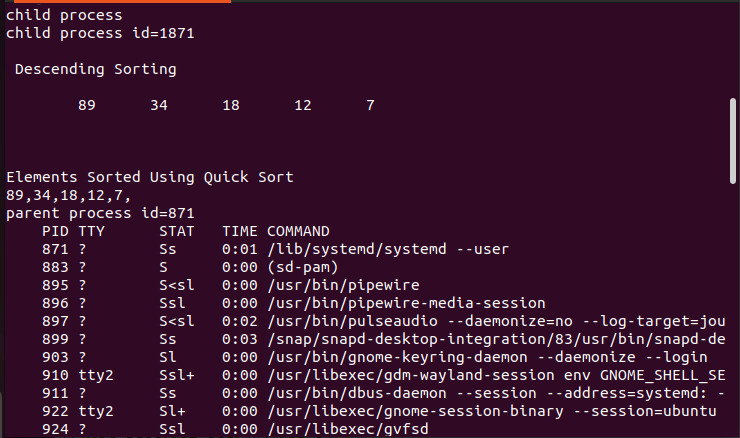
forkeg();

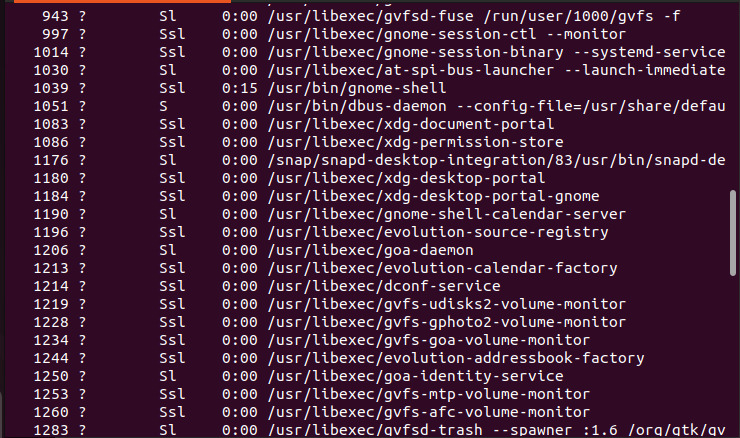
return 0;

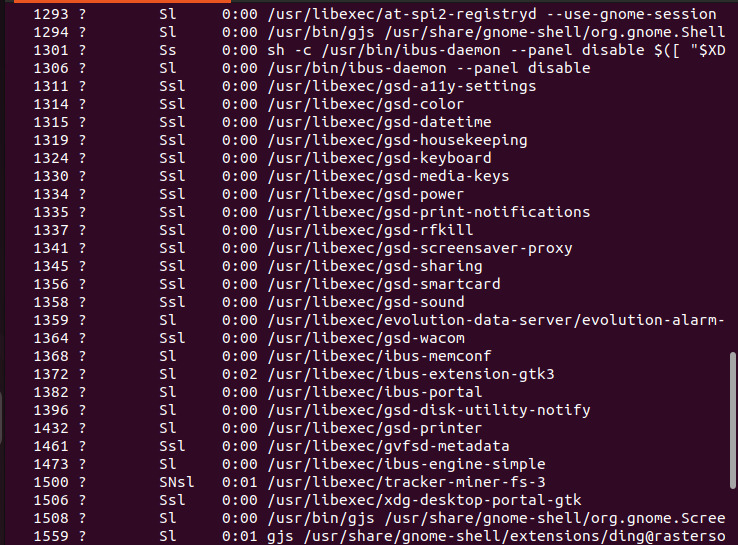
}

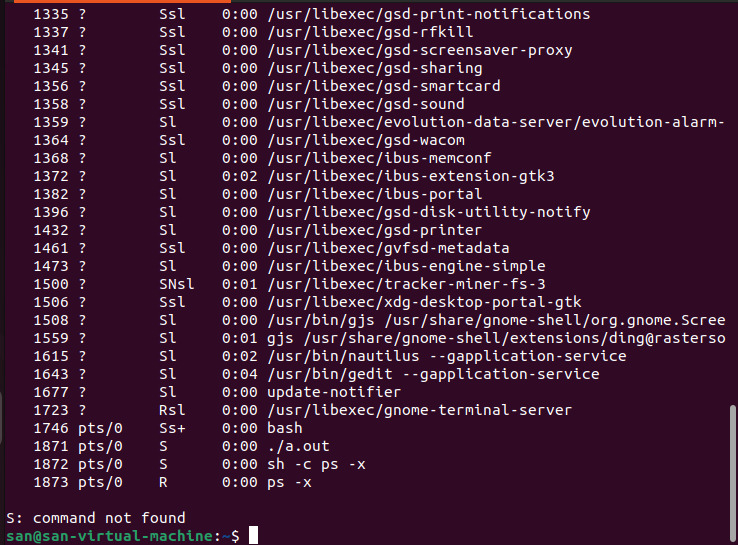
**Output:**

****

****

****

****

****

**Assignment 2-B**

parent.c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <sys/wait.h>

void bubbleSort(int arr[], int n)

{

printf("\nSorting Started\n");

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

for (int j = 0; j < n - 1; j++) {

if (arr[j] > arr[j + 1]) {

int temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

}

printf("\n\nSorting Completed!\n");

}

// To Convert Integer to Char\*

void tostring(char str[], int num) {

int i, rem, len = 0, n;

n = num;

while (n != 0) {

len++;

n /= 10;

}

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

rem = num % 10;

num = num / 10;

str[len - (i + 1)] = rem + '0';

}

str[len] = '\0';

}

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

printf("\nThis is the main process:");

printf("\nProcess Id: %d", getpid());

printf("\nParent Id: %d", getppid());

int arr[] = {10, 5, 1, 60, 20};

int n = 5;

printf("\n\nSorting Array using Bubble Sort:");

bubbleSort(arr, n);

printf("\nForking the current process:");

pid\_t cpid = fork();

if (cpid > 0) {

printf("\n\nParent is Running:\nParentID: %d\nIt's ID: %d\n", getppid(), getpid());

printf("\nParent is waiting for child to complete!\n\n");

wait(NULL);

printf("\n\nParent is Exiting!!\n");

} else if (cpid == 0) {

printf("\n\nChild is running:\nParentID: %d\nIt's ID: %d\n", getppid(), getpid());

char \*arrChar[n + 2];

// Creating ASCII Character Array to Pass as command line Argument

arrChar[0] = (char \*)"child"; // Arg 0 = name of executable file

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

char \*string = (char \*)malloc(sizeof(char) \* 20);

tostring(string, arr[i]);

arrChar[i + 1] = string;

}

arrChar[n + 1] = NULL;

printf("\n\nChild Calling EXECV System Call:\n");

execv("./child", arrChar);

printf("\n\nChild EXECV Call Complete\n");

printf("\n\nChild Execution Complete\n");

// Free allocated memory

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

free(arrChar[i]);

}

} else if (cpid < 0) {

printf("Error");

}

return 0;

}

child.c

#include <stdio.h>

#include <stdlib.h>

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

/\* argv[0] is the program name \*/

if (argc < 2) {

printf("No numbers to process.\n");

return 1;

}

int \*data = (int \*) malloc(argc \* sizeof(int));

if (data == NULL) {

printf("Memory allocation failed.\n");

return 1;

}

printf("\nArgc: %d", argc);

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

data[i] = atoi(argv[i]);

}

// Printing elements in reverse

printf("\nPrinting elements in reverse:");

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

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

}

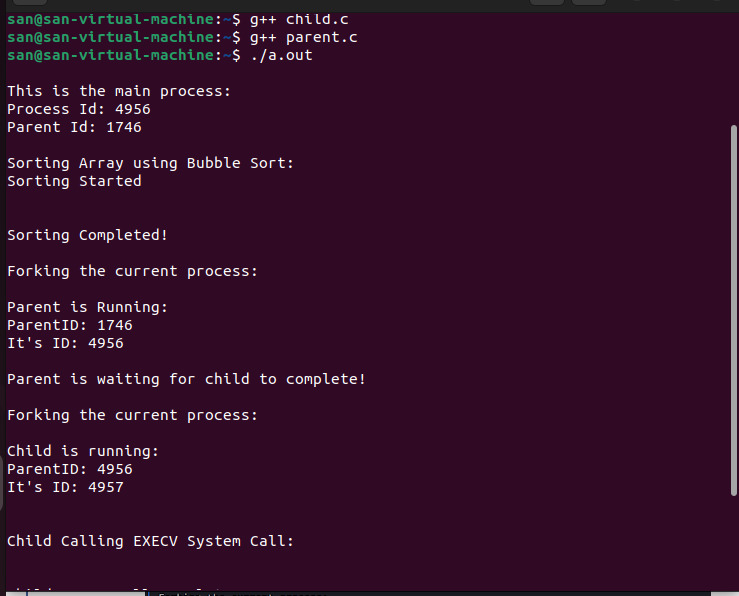
printf("\n\nEXECV task completed\n");

free(data);

return 0;

}

**Output:**

****

****