Name:Akshay S Gadhave
Roll No:221076
SY Comp A3

ASSIGNMENT NO 3.

AIM: Implement following programs to exhibit UNIX Process Control "Program where parent process sorts array elements in ascending order and child process sorts array elements in descending order.Show the demonstration of wait() and zombie process"

THEORY:

Process:-The process is a program in execution or the running instance of a program.

Fork():- Fork system call is used for creating a new process, which is called ***child process***, which runs concurrently with the process that makes the fork() call (parent process). After a new child process is created, both processes will execute the next instruction following the fork() system call. A child process uses the same pc(program counter), same CPU registers, same open files which use in the parent process.

Exec():-It can be used to run a C program by using another C program. It comes under the header file **unistd.h.**
int execvp (const char *file, char *const argv[]);

wait():-It is defined in header <sys/wait.h> & is used to wait till the child process terminates.

Zombie process:-**zombie process** is a **process** that has completed execution (via the exit system call) but still has an entry in the **process** table: it is a **process** in the "Terminated state".

CODE:

Systemcalls.c file:

```
#include<stdio.h>
#include<unistd.h>
```

```
#include<sys/wait.h>
#include<stdlib.h>
#include<string.h>

int main()
{

	char str[20]; //original string array to store int numbers
	char strparent[20]; //string array to sort in parent process
	char ch[3];  //Accept no of elements in this array
	char strchild[20]={}; //string array to sort in child process
	int i,j; //index variables
	int temp;//variable used for swapping
	int status; //to store the pid returned by fork()


	printf("Enter no of elements in array:");
	scanf("%s",ch);


	for(i=0;i<atoi(ch);i++)
	{
		printf("Enter Number:");
		scanf("%s",str);
		strcat(strchild,str);
		strcat(strchild,"-"); //"-" will be used as delimiter while dividing the strings into tokens to
get the actual number
	}

	status=fork();  //fork() system call

	if (status==0) //fork() returns 0 in child process
	{
		char *args[]={"./child",strchild,ch,NULL};  //NULL terminated array of char *
		execvp(args[0],args);
		printf("Ending the child process\n");
	}

	//parent process

	else
	{
		wait(NULL);
		strcpy(strparent,strchild);
		char delimiter[]="-";  //delimiter used in strtok()


		//strtok() used in program actually divides the string into tokens until the specified
delimiter is encountered.
```

```
		/*
		On a first call,the strtok() function expects a C string as argument for str,
		whose first character is used as the starting location to scan for tokens.
		In subsequent calls,the function expects a null pointer
		and uses the position right after the end of the last token as the new starting location for
scanning.
		*/

		char *ptr = strtok(strchild, delimiter);
		int arr[atoi(ch)];
		i=0;
		while(ptr != NULL)
		{
			arr[i++]=atoi(ptr);
			ptr = strtok(NULL, delimiter);
		}

		printf("Array in descending order by parent process:");

		//insertion sort algorithm
		for(i=0;i<atoi(ch);++i)
		{
			j=i;
			while(j>0 && arr[j]>arr[j-1])
			{
				temp=arr[j];
				arr[j]=arr[j-1];
				arr[j-1]=temp;
				j--;
			}
		}

		//print the sorted array
		for(i=0;i<atoi(ch);++i)
			printf("%d  ",arr[i]);

		printf("\n\n");
	}


	return 0;
}
```

child.c file:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main(int argc,char *argv[])
{
	char *strarr=argv[1];
	char delimeter[]="-";

	//strtok() used in program actually divides the string into tokens until the specified 
delimiter is encountered.

	/*
	On a first call,the strtok() function expects a C string as argument for str,
	whose first character is used as the starting location to scan for tokens.
	In subsequent calls,the function expects a null pointer
	and uses the position right after the end of the last token as the new starting 
location for scanning.

	*/

	char *ptr= strtok(strarr,delimeter);  //The first integer number in array
	int arr[atoi(argv[2])]; //argv[2] is the no of elements in array
	int i=0,j=0;  //variables for indexing
	int temp;
	while(ptr != NULL)
	{
		arr[i++]=atoi(ptr);  //converting string to int and then storing it in array
		ptr=strtok(NULL,delimeter);
	}

	printf("\nArray in ascending order by child process: ");

	//insertion sort algorithm

	for(i=0;i<atoi(argv[2]);++i)
	{
		j=i;
		while(j>0 && arr[j]<arr[j-1])
		{
```

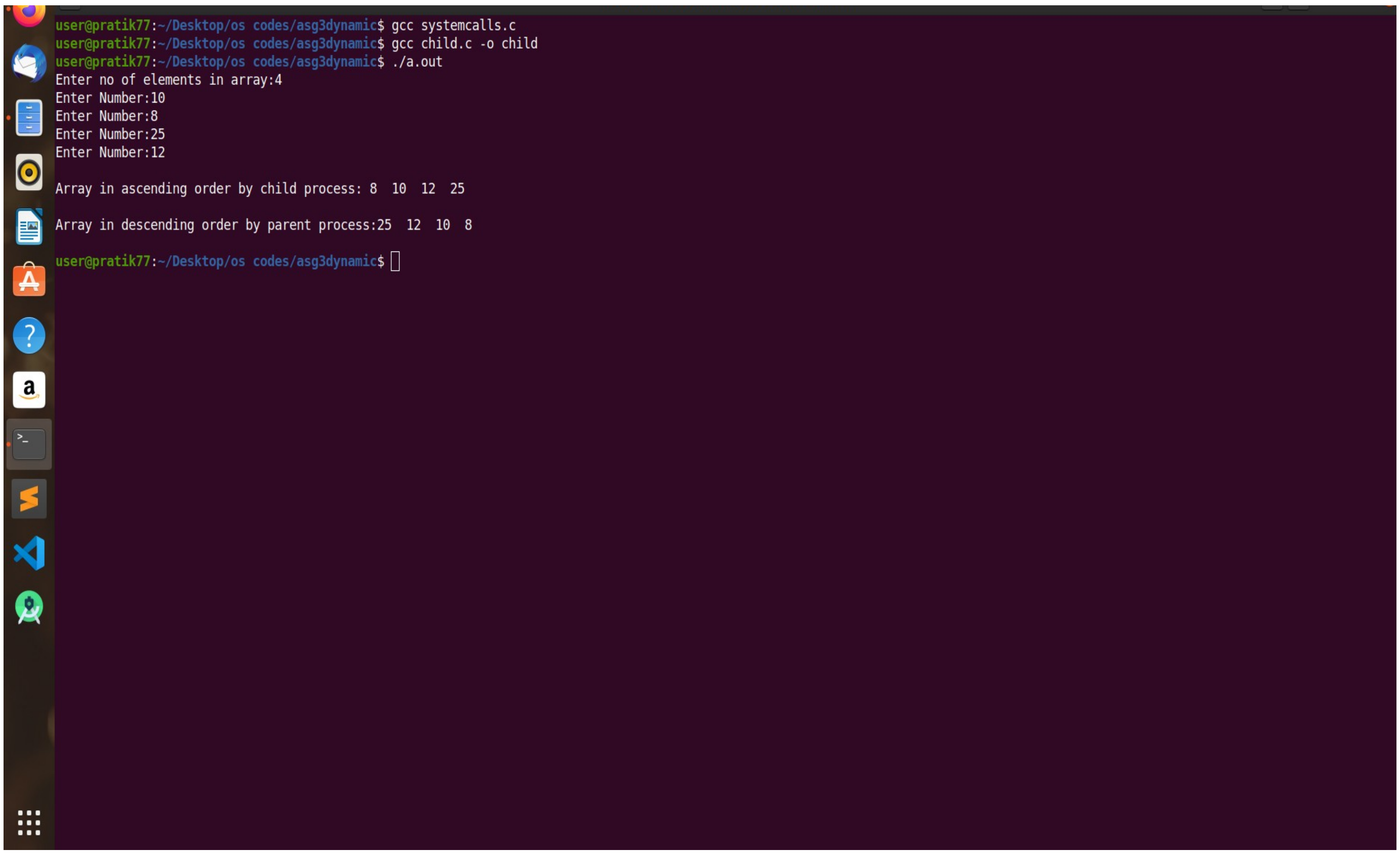
```
                        temp=arr[j];
                        arr[j]=arr[j-1];
                        arr[j-1]=temp;
                        j--;
                }
        }

        //print the sorted array
        for(i=0;i<atoi(argv[2]);++i)
                printf("%d  ",arr[i]);

        printf("\n\n");

        return 0;
}
```

Output Screenshots:



```
user@pratik77:~/Desktop/os codes/asg3dynamic$ gcc systemcalls.c
user@pratik77:~/Desktop/os codes/asg3dynamic$ gcc child.c -o child
user@pratik77:~/Desktop/os codes/asg3dynamic$ ./a.out
Enter no of elements in array:4
Enter Number:10
Enter Number:8
Enter Number:25
Enter Number:12

Array in ascending order by child process: 8  10  12  25

Array in descending order by parent process:25  12  10  8

user@pratik77:~/Desktop/os codes/asg3dynamic$ 
```