UNIVERSIDAD AUTÓNOMA METROPOLITANA
Casa abierta al tiempo
Unidad Iztapalapa

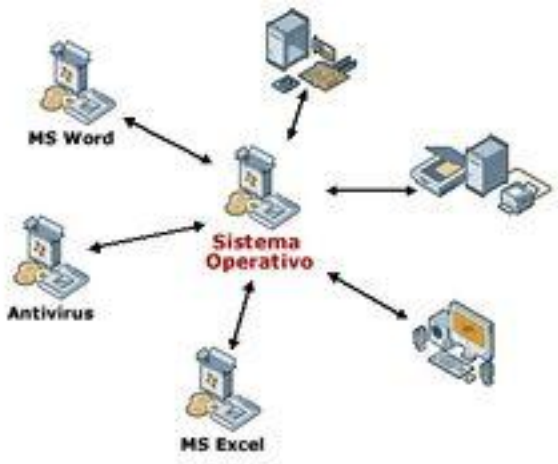
Practica 2. Llamadas al sistema

Sistemas Operativos

Alumno: Hernández Jaimes Luvin Eduardo

Matricula: 2153072883

Profesor: Matadamas Hernández Jorgue



Problemas.

1. Capturar, compilar, ejecutar y entender todos los ejemplos.

Ejemplo 1

```
luvin@luvas:~/Escritorio/Codigos$ ls
1fork.c  2fork.c  3exec.c  4fork-a.c  4fork.c  5fork.c
luvin@luvas:~/Escritorio/Codigos$ gcc 1fork.c
luvin@luvas:~/Escritorio/Codigos$ ./a.out
Soy el proceso padre, valRet = 30082
Soy el proceso hijo, valRet = 0
luvin@luvas:~/Escritorio/Codigos$
```

Ejemplo 2

```
luvin@luvas:~/Escritorio/Codigos$ gcc 2fork.c
luvin@luvas:~/Escritorio/Codigos$ ./a.out
padre: Soy el proceso padre, valRet = 31049
padre: Mi pid es 31048, el pid de mi hijo es 31049
padre: Estoy trabajando ...
hijo: Soy el proceso hijo, valRet = 0
hijo: Mi pid es 31049, el pid de mi padre es 31048
hijo: Estoy trabajando ...
hijo: Dame un valor para regresarlo en el exit: padre: Estoy esperando a que mi hijo termine ...
3
hijo: Listo, bye.
padre: Mi hijo terminó con el valor 3
padre: Listo, bye.
luvin@luvas:~/Escritorio/Codigos$
```

Ejemplo 3

```
luvin@luvas:~/Escritorio/Codigos$ gcc 3exec.c
luvin@luvas:~/Escritorio/Codigos$ ./a.out
1.- execl
2.- execlp
3.- execv
4.- execvp
5.- finalizar proceso padre
Seleccione la función a utilizar: 5
Terminando...
Esperando a que finalicen todos los hijos...
Terminanado
luvin@luvas:~/Escritorio/Codigos$
```

Ejemplo 4

```
luvin@luvas:~/Escritorio/Codigos$ gcc 4fork.c
luvin@luvas:~/Escritorio/Codigos$ ./a.out
Profundidad n = 6
pid=32452, i=1, x=32453
pid=32453, i=2, x=32454
pid=32454, i=3, x=32455
pid=32455, i=4, x=32456
pid=32456, i=5, x=32457
pid=32457, i=6, x=0
luvin@luvas:~/Escritorio/Codigos$ ./a.out
Profundidad n = 2
pid=32491, i=1, x=32492
pid=32492, i=2, x=0
luvin@luvas:~/Escritorio/Codigos$ ./a.out
Profundidad n = 8
pid=32559, i=1, x=32560
pid=32560, i=2, x=32561
pid=32561, i=3, x=32562
pid=32562, i=4, x=32563
pid=32563, i=5, x=32564
pid=32564, i=6, x=32565
pid=32565, i=7, x=32566
pid=32566, i=8, x=0
luvin@luvas:~/Escritorio/Codigos$
```

```
│├─gnome-terminal-─┬─bash─┬─a.out───a.out
││                 │      ├─a.out───a.out───a.out───a.ou+
││                 │      └─a.out───a.out───a.out───a.ou+
```

```
│├─gnome-terminal-─┬─bash─┬─a.out───a.out
││                 │      ├─a.out───a.out───a.out───a.ou+
││                 │      └─a.out───a.out
```

```
│├─gnome-terminal-─┬─bash─┬─a.out───a.out
││                 │      ├─a.out───a.out───a.out───a.ou+
││                 │      └─a.out───a.out───a.out───a.ou+
```

Ejemplo 5

```
luvin@luvas:~/Escritorio/Codigos$ gcc 5fork.c
luvin@luvas:~/Escritorio/Codigos$ ./a.out
Amplitud n = 3
pid=32231, i=1, x=0
pid=32232, i=2, x=0
pid=32230, i=4, x=32233
pid=32233, i=3, x=0
Fueron 3 descendientes
luvin@luvas:~/Escritorio/Codigos$ ./a.out
Amplitud n = 2
pid=32286, i=1, x=0
pid=32285, i=3, x=32287
pid=32287, i=2, x=0
Fueron 2 descendientes
luvin@luvas:~/Escritorio/Codigos$ ./a.out
Amplitud n = 2
pid=32351, i=1, x=0
pid=32350, i=3, x=32352
pid=32352, i=2, x=0
Fueron 2 descendientes
luvin@luvas:~/Escritorio/Codigos$
```

```
│├─gnome-terminal-─┬─bash─┬─a.out───a.out
││                 │      ├─a.out───a.out───a.out───a.ou+
││                 │      └─a.out─┬─a.out
││                 │              ├─a.out
││                 │              └─a.out
```

```
│├─gnome-terminal-─┬─bash─┬─a.out───a.out
││                 │      ├─a.out───a.out───a.out───a.ou+
││                 │      └─a.out─┬─a.out
││                 │              └─a.out
```

```
│├─gnome-terminal-─┬─bash─┬─a.out───a.out
││                 │      ├─a.out───a.out───a.out───a.ou+
││                 │      └─a.out─┬─a.out
││                 │              └─a.out
```

2. Hacer que el árbol del ejemplo 4 (hilera) cuente el número de descendientes del proceso padre, utilizar wait y exit como en el ejemplo 5.

```
luvin@luvas:~/Escritorio/Codigos$ gcc 4fork-a.c
luvin@luvas:~/Escritorio/Codigos$ ./a.out
soy el pid=32776, y tengo 5 desendientes
luvin@luvas:~/Escritorio/Codigos$
```

```
├─gnome-terminal-─┬─bash─┬─a.out───a.out
│                 │      ├─a.out───a.out───a.out───a.ou+
│                 │      └─a.out─┬─a.out
│                 │              ├─a.out
│                 │              ├─a.out
│                 │              ├─a.out
│                 │              └─a.out
```

3. Escribir un programa que genere un árbol de profundidad aleatoria (entre 1 y 5), haciendo que cada proceso tenga entre 1 y 5 hijos. Lo anterior dará un árbol de profundidad menor o igual a cinco en la que los nodos internos del árbol tienen entre 0 y 5 hijos cada uno. Al final hacer que el proceso padre imprima el número de descendientes (Hint: intentarlo primero con un árbol binario de profundidad 5). NOTA: No usar recursividad.

```
luvin@luvas:~/Descargas$ gcc ejemplo3.c
luvin@luvas:~/Descargas$ ./a.out
Soy el proceso 6148
soy el hijo 6149, mi padre es 6148
Soy el proceso 6148
Soy el proceso 6149
soy el hijo 6152, mi padre es 6149
soy el hijo 6151, mi padre es 6148
Soy el proceso 6148
Soy el proceso 6149
soy el hijo 6153, mi padre es 6149
Soy el proceso 6151
soy el hijo 6154, mi padre es 6148
soy el hijo 6155, mi padre es 6151
Soy el proceso 6149
Soy el proceso 6148
Soy el proceso 6151
soy el hijo 6156, mi padre es 6149
soy el hijo 6157, mi padre es 6148
Soy el proceso 6154
soy el hijo 6159, mi padre es 6154
soy el hijo 6158, mi padre es 6151
Soy el proceso 6148
Soy el proceso 6149
Soy el proceso 6151
Soy el proceso 6154
soy el hijo 6162, mi padre es 6151
Soy el proceso 6157
soy el hijo 6163, mi padre es 6154
soy el hijo 6161, mi padre es 6149
soy el hijo 6164, mi padre es 6157
soy el hijo 6160, mi padre es 6148
Soy el proceso 6154
Soy el proceso 6157
luvin@luvas:~/Descargas$ Soy el proceso 6151
Soy el proceso 6160
soy el hijo 6167, mi padre es 6151
soy el hijo 6168, mi padre es 6160
soy el hijo 6166, mi padre es 6157
soy el hijo 6165, mi padre es 6154
Soy el proceso 6154
soy el hijo 6169, mi padre es 6154
Soy el proceso 6157
```

```
Soy el proceso 6157
soy el hijo 6172, mi padre es 6157
Soy el proceso 6160
soy el hijo 6173, mi padre es 6160
Soy el proceso 6160
soy el hijo 6180, mi padre es 6160
luvin@luvas:~/Descargas$
```

4.-Modificar el programa del ejemplo 3 (utilice la función exec que más le agrade) para tener un servidor que, cuando el usuario lo indique, lance una calculadora que pueda sumar, restar, multiplicar y dividir. Al finalizar el programa servidor, debe imprimir el número de calculadoras lanzadas.

```
luvin@luvas:~/Escritorio$ gcc calculadora.c -o cal
luvin@luvas:~/Escritorio$ gcc ejercicio4.c
luvin@luvas:~/Escritorio$ ./a.out
antes del execl soy el papa
despues del execl soy el papa

1.- Suma
2.- Resta
3.- Multiplicación
4.- Division
5.- Finalizar Proceso
Seleccione la función a utilizar: 1

Valor uno: 3

Valor dos: 4

El resultado de la suma es: 7.000000

1.- Suma
2.- Resta
3.- Multiplicación
4.- Division
5.- Finalizar Proceso
Seleccione la función a utilizar: 2

Valor uno: 3

Valor dos: 1

El resultado de la resta es: 2.000000

1.- Suma
2.- Resta
3.- Multiplicación
4.- Division
5.- Finalizar Proceso
Seleccione la función a utilizar: 5
Terminando...
Esperando a que finalicen todos los hijos...
Terminanado
luvin@luvas:~/Escritorio$
```