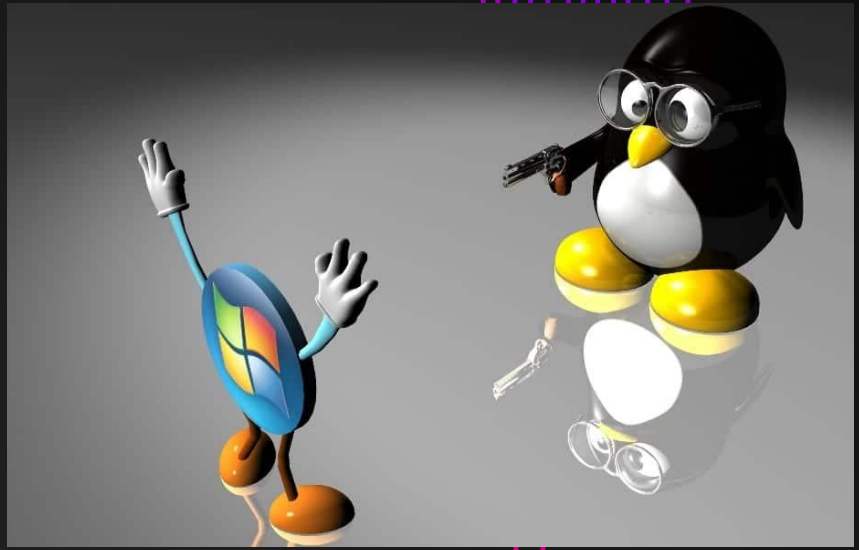
Ayudantia 2: Repaso Control 1

Profesor: Martín Gutiérrez
Ayudante: Dante Hortuvia
Sección 1



Contacto:
Mail: dante.hortuvia@mail.udp.cl
Disc: doshuertos
Telefono:+56922369606

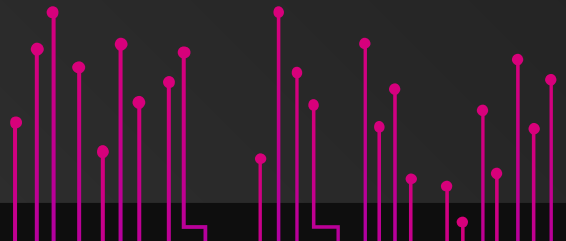
Syscalls

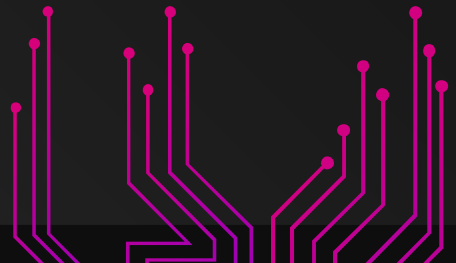
Es un mecanismo donde un programa en modo usuario que solicita recursos y/o servicios al sistema operativo

Fork() : Crea un nuevo proceso duplicando el proceso padre. El proceso hijo tiene el contexto del padre pero tiene su propia copia de memoria y espacio de ejecución, el proceso padre retorna el pid del hijo, mientras el hijo retorna 0.

Exec(): Reemplaza el contenido del proceso actual con un nuevo programa, el proceso mantiene su pid.

Wait(): El proceso padre espera a que todos sus hijos terminen su ejecución para el terminar, esto sirve para sincronizar procesos.





Exit(): Hace que el proceso termine su ejecución retornando un código de su salida. Este libera los recursos utilizados y avisa que terminó.

Sleep(): Hace que el proceso espere una cantidad de segundos a elección para luego continuar con su ejecución

Kill(): Manda una señal a un proceso específico, generalmente para darles termino, algunos de tipos de señal es Sigkill que fuerza la terminación del proceso.

Killall(): Envía señales a todos los procesos que coinciden con un nombre específico. Es útil para terminar múltiples instancias de un programa simultáneamente.

Getpid() = Devuelve el pid del proceso actual.

Fork();

Es una syscall que crea un proceso exacto de el (Hijo), es una copia exacta del proceso padre, con la diferencia que tienen distinto pid y que el hijo tiene un espacio de memoria propia y de ejecución.

Tipos de respuestas:

Si el fork() retorna -1 significa que hubo un error al momento de hacer fork().

Si retorna un número > 0 significa que es el proceso padre

Si retorna un número = 0 significa que es el proceso hijo.

```c
int main(){
    pid_t pid = fork();
    if(pid == 0){
        printf("Hola soy el hijo y mi id es %d y mi pid es %d\n", getpid(), pid);
    }else{

        printf("Hola soy el padre y mi id es %d y mi pid es %d\n",getpid(),pid);

    }
}
```

Ejercicios

Ejercicio 1

```c
int counter = 0; //variable global
int main(){ //codigo1
    pid_t r;
    while(counter < 2){
        r= fork();
        if(r > 0 ){
            execlp("./codigo2", "", NULL);
        }
        counter = counter + 1;

        printf("%d\n", counter);
    }

    return 0;
}
```

```c
int counter = 0; //variable global
int main(){ //codigo2
    pid_t t;
    t = fork();
    if( t > 0 ){
        counter = counter * 2;
    }
    else {
        counter = counter -1;
    }
    printf("%d\n", counter);

    return 0;
}
```

A. ¿Cuántos procesos se crean en total? Justifique.

B. Explique la razón detrás de la existencia de múltiples salidas a partir de la ejecución. Ejemplifique con dos salidas posibles

Ejercicio 2

```c
int main(int argc, char *argv[]){
    if( fork() == fork()){
        printf("Thanos\n");
    }
    else{
        printf("Avengers\n");
    }


    return 0;
}
```

A. ¿Cuántos Procesos se crean?

B. ¿Cual puede ser una salida del código?

Ejercicio 3

```c
int main(){

    pid_t id = fork();
    for(int i = 0; i < 2 ; i++)
    {
        pid_t pid = fork();
        if(pid == id){
            printf("Hola como estas\n");
        }else{
            printf("Adios\n");
        }
    }
}
```

A. Cuántos procesos se crean?

B. Cual puede ser una respuesta del código?

Ejercicio 4

```c
int main() {
    // Codigo 1
    pid_t r[3];

    for (int i = 0; i < 3; i++) {
        r[i] = fork();
        if (r[i] == 0) {
            printf("Ramen\n");
            execlp("./Codigo2", "", NULL);
        }
    }

    sleep(10);
    printf("Sushi\n");

    for (int i = 0; i < 3; i++) {
        kill(r[i], SIGKILL);
    }

    return 0;
}
```

```c
int main() { // Codigo 2
    sleep(666);
    fork();
    printf("Brisket\n");
    exit(0);
}
```

A. Suponga que existe una justa asignación de tiempo de CPU entre los procesos del sistema. ¿Cuántos procesos se crean en total? ¿Cuales son las posibles salidas? Justifique su respuesta.

Ejercicio 5

```c
int num = 1; // variable global

int main() {
    pid_t t = fork();
    if (t != 0) {
        num = num * 2;
        pid_t tt = fork();
        if (tt > 0) {
            num = num - 2;
        } else if (tt < 0) {
            fork();
            num = num + 2;
        }
    }
    num = num + 1;
    sleep(1);
    printf("%d\n", num);
}
```

A. ¿Cuántos procesos se crean en total?

B. Indique dos posibles salidas.

C. Suponga ahora que por una razón misteriosa la primitiva fork() de la línea 6 falla para todos los posibles procesos que ejecutan dicha instrucción. ¿Cual(es) sería(n) la(s) posible(s) salida(s)?

Ejercicio 6

```c
int main() {
    if (fork() != fork()) {
        pid_t t = fork();
        printf("Artorias\n");
        if (t > 0) {
            printf("Ornstein\n");
        } else if (t == 0) {
            printf("Smough\n");
            exit(0);
        } else {
            printf("Hydra\n");
            exit(0);
        }
    }
    printf("Gwyn\n");
}
```

A. ¿Cuántos procesos se crean en total? Justifique su respuesta con un dibujo que represente el árbol de procesos generado.

B. Indique cuántas veces se imprime cada una de las posibles salidas del código.

Control 1 Semestre pasado

Ejercicio 1

```c
int main(){// Proceso X
pit_t r,s,t,p;
r = fork();
s = fork();
t = fork();
p = getpid();
printf("r: %d/n",r);
printf("s: %d/n",s);
printf("t: %d/n",t);

if(p%2 == 0)
{
exit(0);
}
else
{
    execlp("./Y","",NULL);
}
return 0;
}
```

```c
int main(){// Proceso Y
pit_t m,n;
m = getpid();

if(m%2 == 1)
{
exit(0);
}
else
{
    n = fork();
    execlp("./X","",NULL);
}
return 0;
}
```

A. Tome como referencia el primer process id: 4331. Escriba una tabla de posibles valores para r, s y t al ejecutar ./X. Recuerde tomar en cuenta todos los procesos que surgen del primer paso por ./X

B. Mencione cuantos procesos terminan posterior a haber ejecutado código del proceso Y, y justifique detalladamente

C. Explique qué sucedería si se cambia la expresión “m%2 == 1” por “m%2 == 0” en el código del proceso Y y se ejecuta ./X. Describa detalladamente y justifique porque se ejecutan los procesos en la forma en que sucede.

Ejercicio 2

Martín Gutiérrez, profesor de su sección de SOOOOO este semestre, tiene hijos gemelos. Estos bebés deben prepararse, antes de ir a la guardería en la mañana. A su vez, el viejo se prepara y lleva a los niños en auto a la guardería. Considere que los niños toman biberón, se bañan y se visten y que el viejo toma desayuno, se ducha, se viste y prepara el auto para que todos se puedar ir. Luego, lleva a los niños a la guardería y se va al trabajo.

En este contexto descrito, considerando que cada acción es una función y teniendo en cuenta **TODAS** las syscalls referidas a procesos vistas en clases, escriba los códigos de procesos Viejo y Gemelo de acuerdo con la dinámica expuesta.

¡Suerte en su control!