Creación de procesos

fork()

2.1 Creación de procesos

La **creación de un nuevo proceso puede deberse a diversas causas:**

a) La respuesta del sistema operativo frente a la demanda de un servicio (la solicitud por parte de un usuario)
b) A solicitud explicita del usuario, etc.

➢Todos los pasos de creación se llevan a cabo en el modo núcleo.

2.2 La llamada fork

En UNIX los procesos se crean a través de hacer una llamada al sistema con la función ***fork***, esta función se encarga de clonar el proceso que realiza la llamada, es decir provoca la duplicación del proceso actual.

- ✓ El nuevo proceso que se crea recibe una copia del espacio de direcciones del padre.
- ✓ Los dos procesos continúan su ejecución en la instrucción siguiente al fork

Sinytaxis:

```
#include <unistd.h>
pid_t   fork()
```

La llamada fork

Cuando un proceso emite una petición de *fork, el sistema operativo realiza las* siguientes operaciones:

1. Coloca una nueva entrada en la tabla de procesos para el nuevo proceso.

2. Asigna un ID único de proceso al proceso hijo.

3. Hace una copia de la imagen del proceso padre (contexto del padre), a excepción de la memoria compartida.

    a. Copia el contexto del padre al hijo (las variables del padre son copiadas al hijo en espacio de direcciones diferente) y las secciones que son idénticas no se duplican. Sin embargo, los
    b. dos procesos son independientes y su información variará aunque el código sea el mismo, es decir, el hacer una modificación en el hijo, NO se produce en el padre.

La llamada fork

4. Incrementa los contadores de los archivos que pertenecen al padre para indicar que hay un nuevo proceso que también es propietario de esos archivos.

5. Pone al proceso hijo en el estado listo para ejecutar.

6. Devuelve al proceso padre el número de pid del hijo y un valor cero al proceso hijo.

   En caso de fallo:

   - se vuelve un –1 al padre no creándose ningún proceso descendiente y se asigna apropiadamente la variable *errno con valores como:*

     - EAGAIN: Se ha llegado al número máximo de procesos del usuario actual o del sistema.
     - ENOMEN: El núcleo no ha podido asignar suficiente memoria para crear un nuevo proceso.

Crear proceso: fork en UNIX

```
int fork();
```

- Un proceso crea un proceso nuevo. Se crea una relación jerárquica padre-hijo
- El padre y el hijo se ejecutan de forma **concurrente**
- La memoria del hijo se inicializa con una copia de la memoria del padre
  - Código/Datos/Pila
- El hijo inicia la ejecución en el punto en el que estaba el padre en el momento de la creación
  - Program Counter hijo= Program Counter padre
- Valor de retorno del fork es diferente (es la forma de diferenciarlos en el código):
    - Padre recibe el PID del hijo
    - Hijo recibe un 0.

Servicios básicos (UNIX)

Servicio	Llamada a sistema
Crear proceso	fork
Devuelve el PID del proceso	getpid
Devuelve el PID del padre del proceso	getppid
Terminar proceso	exit
Esperar a proceso hijo (**bloqueante**)	wait/waitpid

El hijo hereda las siguientes características del proceso padre:

1. Los ID de usuario y grupo del proceso padre.
2. Los bits de modo usuario y de grupo del proceso padre.
3. La lista de ID de los grupos asociados del proceso padre.
4. Las variables de entorno del proceso padre.
5. Los descriptores de todos los ficheros abiertos por el proceso padre y su desplazamiento.
6. La máscara de modo de creación de fichero del proceso padre.
7. El control de la terminal del proceso padre.
8. El directorio de trabajo actual del padre.
9. Las limitaciones de recursos del proceso padre.

El proceso hijo

1. Tiene un único ID de proceso.

2. Tiene un ID diferente al del proceso padre.

3. El proceso hijo tendrá su propia copia de los descriptores de los ficheros abiertos del padre. El hijo puede cerrar estos ficheros sin afectar al padre. Sin embargo, el padre y el hijo comparten el desplazamiento para cada descriptor lo que significa que, si ambos escriben en el fichero a la vez, la salida será entremezclada. Además, si los dos leen del fichero, cada un recibirá parte de los datos.

4. El proceso hijo no tendrá ninguno de los bloqueos de ficheros que su padre haya creado.