

**INSTITUTO POLITÉCNICO NACIONAL.**

ESCUELA SUPERIOR DE CÓMPUTO.

**SISTEMAS OPERATIVOS.**

**PRÁCTICA 3**

CREACIÓN DE PROCESOS.

**Integrantes del equipo:**

* Chavarría Vázquez Luis Enrique.
* Juárez Espinoza Ulises.
* Machorro Vences Ricardo Alberto.
* Pastrana Torres Víctor Norberto.



Grupo 2CM6

Índice de contenido.

[**Glosario de términos.** 1](#_Toc54148444)

[**Procesos.** 1](#_Toc54148445)

[**Ejecución en 1er plano.** 1](#_Toc54148446)

[Proceso iniciado por el usuario o interactivo. 1](#_Toc54148447)

[**Ejecución en 2o plano.** 1](#_Toc54148448)

[**Proceso zombi.** 1](#_Toc54148449)

[Proceso parado que queda en la tabla de procesos hasta que termine su padre. Este hecho se produce cuando el proceso padre no recoge el código de salida del proceso hijo. 1](#_Toc54148450)

[**Proceso huérfano.** 1](#_Toc54148451)

[Proceso en ejecución cuyo padre ha finalizado. 1](#_Toc54148452)

[**Contenido** 2](#_Toc54148453)

[**Pr** 2](#_Toc54148454)

[**Conclusiones.** 3](#_Toc54148455)

[**Chavarría Vázquez Luis Enrique.** 3](#_Toc54148456)

[**Juárez Espinoza Ulises.** 3](#_Toc54148457)

[**Machorro Vences Ricardo Alberto.** 3](#_Toc54148458)

[**Pastrana Torres Victor Norberto.** 3](#_Toc54148459)

[Bibliografía 4](#_Toc54148460)

Índice de figuras

**No se encuentran elementos de tabla de ilustraciones.**

Índice de tablas

**No se encuentran elementos de tabla de ilustraciones.**

# **Glosario de términos.**

## **Procesos.**

No es más que un programa o comando en ejecución. Ahora vale la pena ahondar en algunas características.

* Un proceso consta de código, datos y pila.
* Los procesos existen en una jerarquía de árbol (varios Hijos, un sólo padre).
* El sistema asigna un identificador de proceso (PID) único al iniciar el proceso.
* El planificador de tareas asigna un tiempo compartido para el proceso según su prioridad (sólo root puede cambiar prioridades).

## **Ejecución en 1er plano.**

Proceso iniciado por el usuario o interactivo.

## **Ejecución en 2o plano.**

Proceso no interactivo que no necesita ser iniciado por el usuario.

## **Proceso zombi.**

Proceso parado que queda en la tabla de procesos hasta que termine su padre. Este hecho se produce cuando el proceso padre no recoge el código de salida del proceso hijo.

## **Proceso huérfano.**

Proceso en ejecución cuyo padre ha finalizado.

# **Contenido (Investigación)**

## **Creación de procesos.**

### Subrutina fork:

Crea un nuevo proceso (hijo), copia casi exacta del proceso generador (padre).

- Formato:

#include <unistd.h>

pid\_t fork ();

- Devuelve:

0 al proceso hijo y PID del hijo al proceso padre (-1, si error).

- Comentarios:

a) La versión BSD (en la librería libbsd.a) es:

int vfork ();

b) Atributos que hereda el proceso hijo.

Entorno.

Bit FD\_CLOEXEC para cada descriptor de fichero.

Señales capturadas.

SUID y SGID.

Estado de privilegios y prioridades.

Librerías compartidas y segmentos de memoria compartida.

PGID y TTYGID.

Directorio actual y directorio raíz.

Máscara y límites de medida para ficheros.

Eventos y estado de auditoría.

Estado de depuración.

c) Atributos diferenciadores entre padre e hijo:

PID único.

PPID distintos (el PPID del hijo coincide con el PID del padre).

El proceso hijo tiene su propia copia de los descriptores de fichero del padre, pero comparte con éste un puntero a fichero para cada descriptor del proceso padre.

Bloqueos de proceso, texto y datos no se heredan.

Las subrutinas times se ponen a 0.

Las alarmas pendientes toman su valor inicial.

Se eliminan las señales pendientes para el proceso hijo.

- Ejemplos:

/\* fork.c - Ejecución conjunta de procesos padre e hijo \*/

#include <stdio.h>

#include <unistd.h>

main ()

{

printf ("Ejemplo de fork.\n");

printf ("Inicio del proceso padre. PID=%d\n", getpid ());

if (fork() == 0)

{ /\* Proceso hijo \*/

printf ("Inicio proceso hijo. PID=%d, PPID=%d\n",

getpid (), getppid ());

sleep (1);

}

else

{ /\* Proceso padre \*/

printf ("Continuación del padre. PID=%d\n", getpid ());

sleep (1);

}

printf ("Fin del proceso %d\n", getpid ());

exit (0);

}

Ejemplo de fork.

Inicio proceso padre. PID=8153

Inicio proceso hijo. PID=6618, PPID=8153

Continuación proceso padre. PID=8153

Fin del proceso 6618

Fin del proceso 8153

- Ejemplo:

/\* fork\_huerf.c - Ejemplo de proceso huérfano \*

#include <stdio.h>

#include <unistd.h>

main ()

{

printf ("Ejemplo de proceso huérfano.\n");

printf ("Inicio del proceso padre. PID=%d\n", getpid ());

if (fork () == 0)

{

printf ("Inicio proceso hijo. PID=%d, PPID=%d\n",

getpid (), getppid ());

sleep (1);

printf ("El proceso queda huérfano. PID=%d PPID=%d\n",

getpid (), getppid ());

}

else

printf ("Concinuación del padre. PID=%d\n", getpid ());

printf ("Fin del proceso %d\n", getpid ());

exit (0);

}

Ejemplo de proceso huérfano.

Inicio proceso padre. PID=11330

Inicio proceso hijo. PID=6467, PPID=11330

Continuación proceso padre. PID=11330

Fin del proceso 11330

$punto indicativo> El proceso queda huérfano. PID=6467, PPID=1

Fin del proceso 6467

Notas:

En el ejemplo, el proceso padre no espera la finalización del proceso hijo y termina antes que éste.

Cuando un proceso queda huérfano, el proceso de iniciación (init) se convierte en su padre.

Una vez que finaliza el proceso padre, se devuelve el control al intérprete de comandos, de ahí que aparezca el mensaje del "punto indicativo". El proceso hijo no deberá mandar mensajes a la consola, como ocurre en este ejemplo.

1,3,2. Subrutinas wait y waitpid:

- Descripción:

Espera a que pare o termine un proceso hijo, permitiendo obtener sus estados de salida.

Una señal no bloqueada o no ignorada puede reactivar el proceso padre.

- Formato:

#include <sys/wait.h>

pid\_t wait (estados)

int \*estados;

pid\_t wait ((void \*) 0);

pid\_t waitpid (PID, estados, opciones)

pid\_t PID;

int \*estados, opciones;

- Parámetros:<

PID - PID del proceso o grupo de proceso. Sus valores son:

-1: waitpid actúa igual que wait, esperando cualquier hijo.

>0: PID de un proceso hijo determinado.

0: para cualquier hijo con el mismo grupo de procesos que el padre.

<-1: para cualquier hijo cuyo grupo de proceso sea igaul al valor absoluto de PID.

opciones - Máscara de opciones. Sus bits son:

WNOHANG: evita la suspensión del padre mientras esté esperando a algún hijo.

WUNTRACED: el padre obtiene información adicional si el hijo recibe alguna de las señales SIGTTIN, SIGTTOU, SIGSSTP o SIGTSTOP .

estados - Puntero a una tabla con los estados de salida de los procesos.

- Devuelve:

0, si no ha terminado ningún proceso.

- Macros:

WIFSTOPPED (estado) /\* !=0, si estado es de un hijo parado \*/

pid\_t estado;

int WSTOPSIG (estado) /\* Nú de señal que ha causado la parada \*/

pid\_t estado;

WIFEXITED (estado) /\* !=0, si estado es de salida normal \*/

pid\_t estado;

int WEXITSTATUS (estado) /\* 8 bits bajos del estado de salida \*/

pid\_t estado;

WIFSIGNALED (estado) /\* !=0, si estado es de salida anormal \*/

pid\_t estado;

int WTERMSIG (estado) /\* Nú de sañal que ha causado la slida \*/

pid\_t estado;

- Cometarios:

a) Estas subrutinas pueden verse afectadas por la señal SIGCHLD (ver sigaction).

b) La subrutina wait espera la terminación de cualquier proceso hijo.

- Ejemplos:

/\* waitpid.c - Esperar la terminación de un proceso hijo \*/

#include <stdio.h>

#include <signal.h>

#include <sys/wait.h>

main ()

{

pid\_t id\_padre; /\* PID del proceso padre \*/

pid\_t id\_hijo; /\* PID del proceso hijo \*/

int estado; /\* Estado de salida \*/

printf ("Ejemplo de waitpid.\n");

printf ("Inicio proceso padre. PID=%d\n", getpid ());

id\_padre = getpid ();

if ((id\_hijo = fork ()) == 0)

{ /\* Proceso hijo \*/

printf ("Inicio proceso hijo. PID=%d, PPID=%d\n",

getpid (), id\_padre);

sleep (3);

printf ("Salida proceso hijo. PID=%d\n", getpid ());

exit (getpid () > id\_padre); /\* 1, si PID > PPID \*/

}

else

{

signal (SIGINT, SIG\_IGN); /\* Ignorar CTRL-C \*/

while (waitpid (id\_hijo, &estado, 0) != id\_hijo);

if (WIFSIGNALED (estado))

printf ("El proceso hijo ha recibido la señal %d\n", WTERMSIG (estado));

if (WIFEXITED (estado))

{

printf ("Estado de salida del proceso hijo: %d\n", WEXITSTATUS (estado));

if (WEXITSTATUS (estado) == 1)

printf ("PID hijo > PID padre.\n");

else

printf ("PID padre > PID hijo.\n");

}

printf ("Fin del proceso %d\n", getpid ());

exit (0);

}

Ejemplo de waitpid.

Inicio proceso padre. PID=24213

Inicio proceso hijo. PID=31638, PPID=24213

Fin proceso hijo. PID=31638

Estado de salida del proceso hijo: 1

PID hijo > PID padre

Fin del proceso 24213

La salida siguiente muestra el efecto de generar una señal de interrupción pulsando [CTRL][C]. Dicha señal provoca la terminación automática del proceso hijo, mientras que el proceso padre la ignora (ver signal).

Ejemplo de waitpid.

Inicio proceso padre. PID=7240

Inicio proceso hijo. PID=5705, PPID=7240

^CEl proceso hijo ha recibido la señal: 2

Fin del proceso 7240

# **Conclusiones.**

## **Chavarría Vázquez Luis Enrique.**

## **Juárez Espinoza Ulises.**

## **Machorro Vences Ricardo Alberto.**

## **Pastrana Torres Victor Norberto.**

# Bibliografía

**No hay ninguna fuente en el documento actual.**