
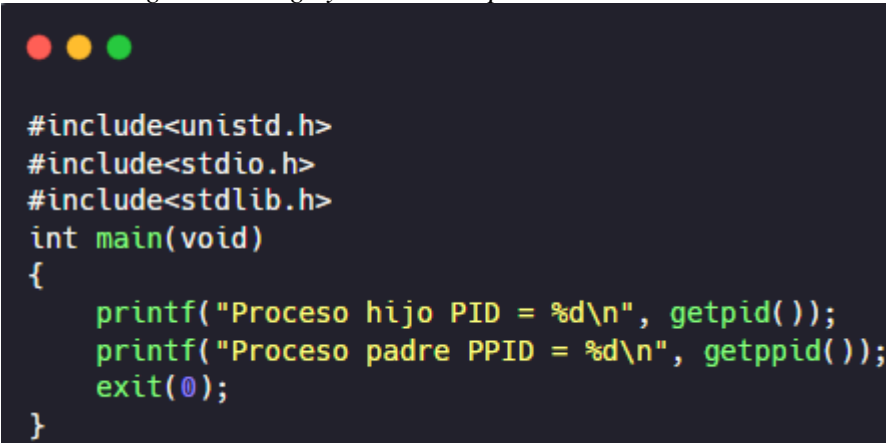
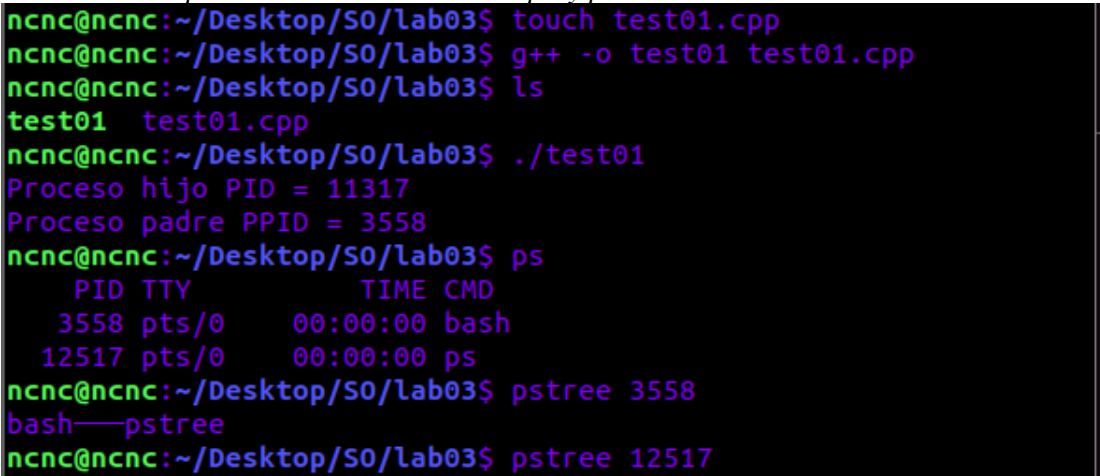


	<p align="center">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
Aprobación: 2022/03/01	Código: GUIA-PRLE-001	Página: 1

INFORME DE LABORATORIO

INFORMACIÓN BÁSICA					
ASIGNATURA:	Sistema Operativos				
TÍTULO DE LA PRÁCTICA:					
NÚMERO DE PRÁCTICA:	03	AÑO LECTIVO:	2022-B	NRO. SEMESTRE:	VI (sexto)
FECHA DE PRESENTACIÓN	12/10/2022	HORA DE PRESENTACIÓN	23:59		
INTEGRANTE (s): Yoset Cozco Mauri				NOTA:	
DOCENTE(s): <i>ROLANDO JESUS CARDENAS TALAVERA</i>					

SOLUCIÓN Y RESULTADOS	
I. ACTIVIDADES	
1. Analice el siguiente código y de una interpretación del resultado obtenido de acuerdo con el marco teórico.	
 <pre> #include<unistd.h> #include<stdio.h> #include<stdlib.h> int main(void) { printf("Proceso hijo PID = %d\n", getpid()); printf("Proceso padre PPID = %d\n", getppid()); exit(0); } </pre>	<p>Revision de los procesos mediante comando <code>ps</code> y <code>pstree</code>:</p>  <pre> ncnc@ncnc:~/Desktop/SO/lab03\$ touch test01.cpp ncnc@ncnc:~/Desktop/SO/lab03\$ g++ -o test01 test01.cpp ncnc@ncnc:~/Desktop/SO/lab03\$ ls test01 test01.cpp ncnc@ncnc:~/Desktop/SO/lab03\$./test01 Proceso hijo PID = 11317 Proceso padre PPID = 3558 ncnc@ncnc:~/Desktop/SO/lab03\$ ps PID TTY TIME CMD 3558 pts/0 00:00:00 bash 12517 pts/0 00:00:00 ps ncnc@ncnc:~/Desktop/SO/lab03\$ pstree 3558 bash──pstree ncnc@ncnc:~/Desktop/SO/lab03\$ pstree 12517 </pre>

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 2</p>

El método *getpid()*: Nos muestra el ID de un proceso padre.

El método *getppid()*: Nos muestra el ID de un proceso hijo.

En este caso tenemos el proceso padre *bash* con el PID: 3558

Y otro proceso hijo con PID: 11317:

Adicional a eso al listar procesos con el comando *ps*, se inicia este proceso hijo llamado *ps* con PID: 12517.

II. SOLUCIÓN DE EJERCICIOS/PROBLEMAS

EJERCICIOS PROPUESTOS

- El siguiente código crea un proceso hijo, realice un seguimiento de la variable *value* y describa el porque tiene ese comportamiento.



```
#include <sys/types.h>
#include <sys/wait.h>
#include <stdio.h>
#include <unistd.h>

int value = 5;
int main()
{
    pid_t pid;
    pid = fork();
    if (pid == 0) { /* child process */
        value += 15;
        return 0;
    }
    else if (pid > 0) { /* parent process */
        wait(NULL);
        printf("PARENT: value = %d", value); /*LINE A*/
        return 0;
    }
}
```

```
ncnc@ncnc:~/Desktop/S0/lab03$ g++ -o prop_ejer01 prop_ejer01.cpp
ncnc@ncnc:~/Desktop/S0/lab03$ ./prop_ejer01
```

```
PARENT: value = 5ncnc@ncnc:~/Desktop/S0/lab03$ █
```

Value empieza con un valor de 5, *pid_t* crea un nuevo proceso, *fork()* crea un proceso duplicado, siendo el *pid* actualmente de valor 0, entra en la condicional para crear un proceso hijo, aumentando el valor de *value* en 15, valiendo 20, ahora el *pid* después de su creación tiene asignado un nuevo valor PID, reiniciando el valor de *value* a su valor original 5, mostrando así en la impresión 5(valor inicial).

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 3</p>

- En el siguiente código, detalle que parte del código es ejecutada por el proceso padre y que porción del código es ejecutada por el proceso hijo. Describa la actividad de cada uno.

```

#include <sys/types.h>
#include <stdio.h>
#include <sys/wait.h>
#include <unistd.h>

int main(void)
{
    pid_t pid;
    /* fork a child process */
    pid = fork();
    if (pid < 0) { /* error occurred */
        fprintf(stderr, "Fork Failed");
        return 1;
    }
    else if (pid == 0) { /* child process */
        execlp("/bin/ls", "ls", NULL);
    }
    else { /* parent process */
        /* parent will wait for the child to complete */
        wait(NULL);
        printf("Child Complete \n");
    }
    return 0;
}

```

```



ncnc@ncnc:~/Desktop/S0/lab03$ g++ -o prop_ejer02 prop_ejer02.cpp
ncnc@ncnc:~/Desktop/S0/lab03$ ./prop_ejer02
prop_ejer01      prop_ejer02.cpp  prop_ejer05_consumer.cpp  test01.cpp
prop_ejer01.cpp  prop_ejer03.cpp  prop_ejer05_producer.cpp
prop_ejer02      prop_ejer04.cpp  test01
Child Complete
ncnc@ncnc:~/Desktop/S0/lab03$

```

El proceso padre es iniciado en `pid_t pid`, a partir de esto se crea un `fork()` para iniciar al proceso hijo, este proceso llama al proceso `ls` del sistema, una vez ejecutado este proceso el padre acaba la espera e imprime "child complete".

III. SOLUCIÓN DEL CUESTIONARIO

- ¿Cuál es la principal característica de crear un proceso utilizando la función FORK?
- ¿Cuántos procesos FORK se pueden crear de forma secuencial? ¿Existe algún límite establecido por el sistema operativo?
- ¿Qué trabajo realiza la función EXECLP? Explique utilizando un ejemplo de utilización del comando.

	<p align="center">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p align="center">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 4</p>

IV. CONCLUSIONES

RETROALIMENTACIÓN GENERAL

REFERENCIAS Y BIBLIOGRAFÍA