Generador de Código de Estructuras Jerárquicas de Procesos Emparentados en Ejecución Concurrente

Cherencio, Guillermo Ruben; Romero, Juan Carlos; Perello, Mario Gerardo
Universidad Tecnológica Nacional, Facultad Regional Delta

Abstract

En el contexto de la enseñanza de Sistemas Operativos, la creación de varios procesos emparentados implica para el alumno un desafío cognitivo. Hasta este momento, el alumno sólo había experimentado con la programación y ejecución de un único proceso, pero ahora se encuentra con la problemática de la creación de un proceso que crea procesos emparentados, que se ejecutan concurrentemente y además pueden tener necesidades de comunicación y sincronización entre ellos. A partir de analizar ejercicios de creación de procesos, se tipificaron 4 casos de estructuras jerárquicas de procesos emparentados. Se realizó el desarrollo de un software, que permite generar para cada caso el código en el lenguaje de programación C. El software desarrollado le permite al alumno generar el código, modificarlo, compilarlo y ejecutarlo. Con esta herramienta se pretende ayudar al alumno a superar este desafío cognitivo, facilitar el proceso de enseñanza – aprendizaje de la asignatura Sistemas Operativos; en la comprensión de temas tales como: creación de procesos, comunicación y sincronización de procesos, planificación de procesos. La herramienta desarrollada es de uso libre y se encuentra publicada bajo licencia LPGL.

Palabras Clave
Proceso fork Linux Unix C

1. Introducción

En la Asignatura Sistemas Operativos tenemos una unidad curricular dedicada al estudio de los procesos. En esta unidad es notoria la dificultad de los alumnos para comprender todo lo relacionado con la creación de procesos, utilizando la llamada al sistema *fork*[1].

Consideramos que esta dificultad deriva del hecho de que el alumno tiene – hasta el momento- una visión de la programación centrada en un único proceso que resuelve un problema determinado. Se trata de la implementación de un algoritmo[2], expresado en algún lenguaje de programación, que más tarde se convertirá en un proceso, el cual describe una traza de instrucciones que tiene un principio y un fin. Su esfuerzo se centra en la correcta implementación de un algoritmo y que la salida del proceso sea la esperada.

La asignatura sistemas operativos propone un cambio de paradigma en la gestión de procesos hasta ahora estudiada por el alumno y el cambio es realmente radical, introduciendo varias dificultades en la programación:

- un proceso es capaz de crear otros procesos idénticos a él, estableciéndose entre ellos una relación jerárquica de tipo padre - hijo
- el proceso creado no es algo estático, sino que el mismo puede comenzar su ejecución ni bien lo disponga el planificador del sistema operativo[3], a partir de la línea de código siguiente a la llamada *fork*
- el proceso creado es copia del proceso creador, incluyendo la información de contexto[4] y sus datos
- la resolución de ciertos problemas pueden requerir comunicación y sincronización entre los procesos[5] creados
- La planificación de ejecución del proceso depende del sistema

operativo y está fuera del control del programador, el orden de finalización de los procesos no está determinado por el programador, en primera instancia

- Los procesos cambian de estado durante la ejecución y pueden pasar a estado *huérfano* o *zombie* [6]

Ahora el alumno tiene que lidiar con varios procesos, que describen distintas trazas de ejecución y emiten distintas salidas en las cuales debe concentrarse. Y todo ello ocurre de forma concurrente, a esto nos referimos como "desafío cognitivo".

Analizando los ejercicios evaluados en la asignatura en los últimos 10 años referidos a este tema, hemos podido tipificar una serie de casos a los que nos referimos en la siguiente sección.

2. Descripción de Casos

Como se mencionó en el punto anterior, la creación de procesos implica relaciones jerárquicas de tipo padre – hijo, por lo tanto, al tipificar los distintos ejercicios evaluados sobre este tema, podemos enmarcarlos en distintas figuras jerárquicas, las cuales las hemos denominado como: "Creación jerárquica de procesos (caso 1)", "Creación de procesos en abanico (caso 2)", "Creación de procesos en forma de árbol balanceado (caso 3)", "Creación lineal de procesos (caso 4)".

Los casos se graficarán a través de diagramas de procesos en donde cada nodo representa un proceso. Una línea que conecta un nodo con otro representa una relación jerárquica padre – hijo, los diagramas se leen de arriba hacia abajo, los procesos hijos se grafican debajo de los procesos padre. Siempre comenzamos con un nodo raíz, que representa el proceso principal que fue ejecutado desde el intérprete de comandos del sistema operativo (*shell*)[7] que denominamos arbitrariamente como "main". El resto de los procesos los denominamos como p1, p2,.. pN; en donde los nombres de los procesos son unívocos, de esta forma, le damos un nombre lógico a cada proceso que será utilizado por el generador de código y le permitirá al alumno identificar el proceso dentro del código generado. A continuación se describen cada uno de estos casos.

2.1 Creación jerárquica de procesos (caso 1)

Este caso describe a todos los ejercicios en donde se le solicita al alumno una relación jerárquica libre y arbitraria de procesos.

Bajo este caso hemos agrupado a todos los ejercicios que no coinciden con las formas jerárquicas expresadas en los otros casos:

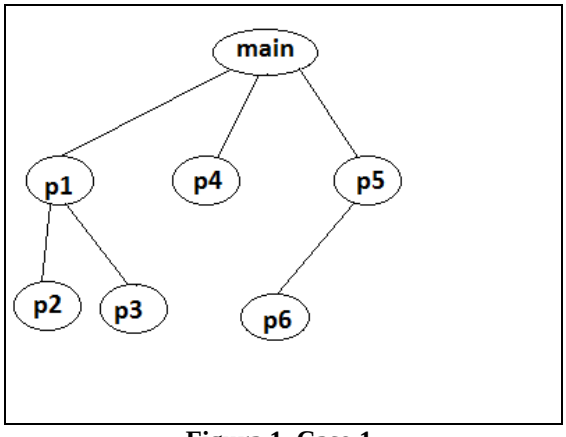

Figura 1. Caso 1.

2.2 Creación de procesos en abanico (caso 2)

Este caso describe a todos los ejercicios en donde se le solicita al alumno la creación de N procesos hijos, en donde todos ellos son hermanos, hijos del proceso "main". Podríamos decir que este es un caso particular del caso 1:

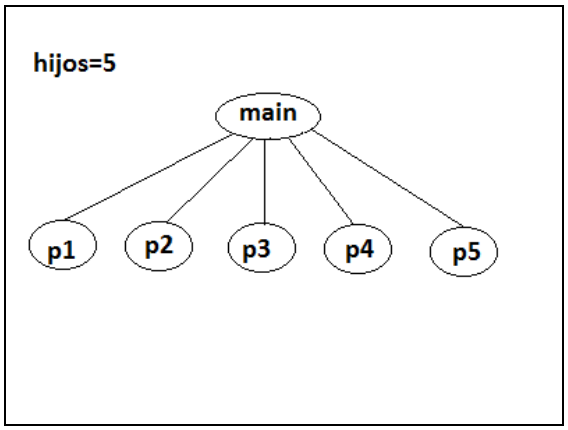

Figura 2. Caso 2.

2.3 Creación de procesos en forma de árbol balanceado (caso 3)

Este caso describe a todos los ejercicios en donde se le solicita al alumno la creación de procesos bajo una forma de árbol balanceado, en donde cada nodo o proceso tiene un número fijo de procesos hijos y a su vez, el árbol tiene un número fijo de niveles. Podríamos decir que este es un caso particular del caso 1:

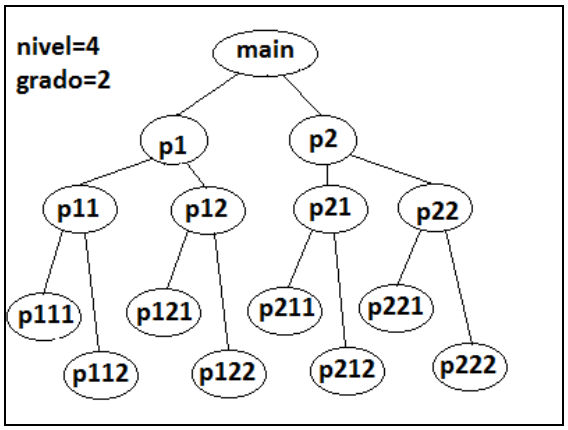

Figura 3. Caso 3.

2.4 Creación lineal de procesos (caso 4)

Este caso describe a todos los ejercicios en donde se le solicita al alumno la creación de N procesos, en donde cada proceso tiene un solo hijo (excepto el último proceso). Podríamos decir que este es un caso particular del caso 1:

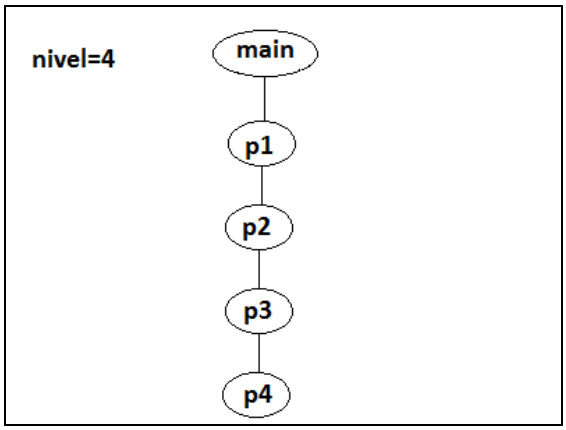

Figura 4. Caso 4.

3. Análisis de Casos

El objetivo de este trabajo es crear un software educativo que pueda ser utilizado por el alumno para generar el código, compilar y ejecutar cada uno de los cuatros casos presentados. El software tendrá una interfaz gráfica y está orientado a sistemas operativos que dispongan de la llamada al sistema *fork* para la creación de procesos.

En cada diagrama -en el ángulo superior izquierdo- se ha indicado para los casos 2, 3 y 4 los parámetros que deberán ser ingresados como entrada del proceso de generación de código. Para el caso 1, hemos considerado que el parámetro de entrada es la estructura jerárquica en sí misma.

En el análisis de los casos también debemos considerar qué tarea realizará cada proceso creado, se han identificado 3 posibles tareas a realizar que describimos a continuación:

- **“while(1);”** : los procesos emitirán un mensaje por pantalla indicando su nombre lógico único, el número de proceso (*process id, PID*) de su proceso padre, su propio número de proceso y luego de ello, ingresarán en un loop infinito (*while(1);*) que impedirá la finalización del proceso, pero ello permitirá que el alumno visualice correctamente en la consola todos los mensajes enviados por los procesos
- **“wait();”**: además de emitir los mensajes de identificación de proceso, también se le solicita al alumno una sincronización mínima entre procesos, de manera tal, que no quepa la posibilidad de generar procesos *zombies* ni *huérfanos*. Esta sincronización permitirá establecer un orden de finalización de procesos, en donde el proceso “main” será el último en finalizar
- **“exec()**;**”**:se le solicita al alumno que cada proceso creado se transforme en otro proceso, a través de otra llamada al sistema (familia de funciones exec)[8], es posible transformar al proceso creado en otro proceso

Para el caso 1, es posible solicitar al alumno “while(1);”, “wait();” y “exec();”.

Para el caso 2, es posible solicitar al alumno “while(1);”, “wait();” y “exec();”.

Para el caso 3, es posible solicitar al alumno “while(1);”, “wait();” pero no es lógicamente posible solicitar “exec();”.

Para el caso 4, es posible solicitar al alumno “while(1);”, “wait();” pero no es lógicamente posible solicitar “exec();”.

Las combinaciones entre los casos y el tipo de tarea a realizar por los procesos, deben considerarse en el diseño del software propuesto.

3.1 **Desarrollo del software propuesto**

El software se desarrolló en lenguaje de programación Java®[9] y se lo denominó como “Fork Generator”, la interfaz gráfica se desarrolló con el framework Java Swing[10]; es muy simple de utilizar, el usuario debe seleccionar el caso, ingresar los parámetros de entrada del mismo, seleccionar la tarea a realizar por cada proceso creado y luego presionar el botón “C” para generar el código en el lenguaje de programación C[11]. El código generado se mostrará en un área de texto que le permitirá al alumno visualizarlo o modificarlo. Luego se puede presionar el botón “!” para lanzar la compilación y ejecución del código que se realizará en una nueva terminal para que el alumno pueda apreciar la salida de todos los procesos en una única ventana:

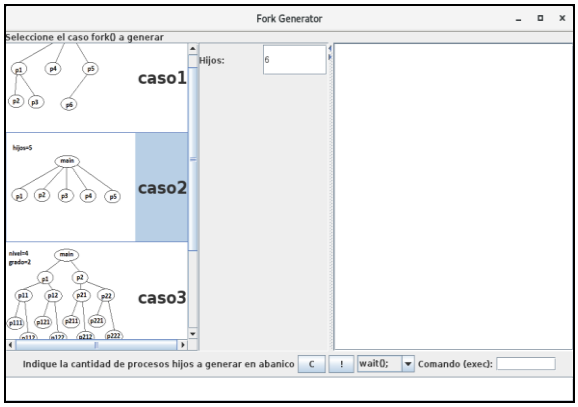

Figura 5. Interfaz Gráfica Fork Generator.

Para el caso 1 se utilizó el componente Swing JTree[12] permitiendo al usuario agregar los procesos en la posición jerárquica que desee utilizando las teclas “+” para agregar y “-“ para quitar:

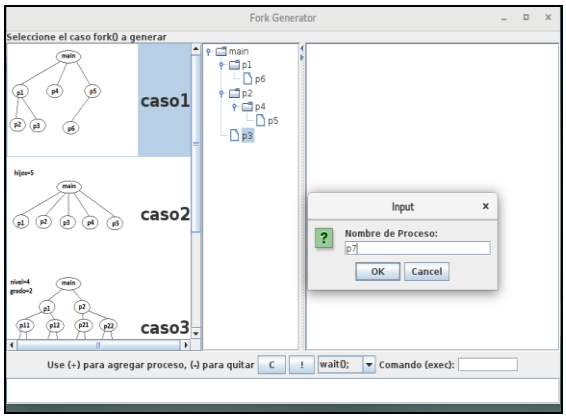

Figura 6. Fork Generator, Caso 1.

Aquí podemos apreciar la ejecución de un caso:

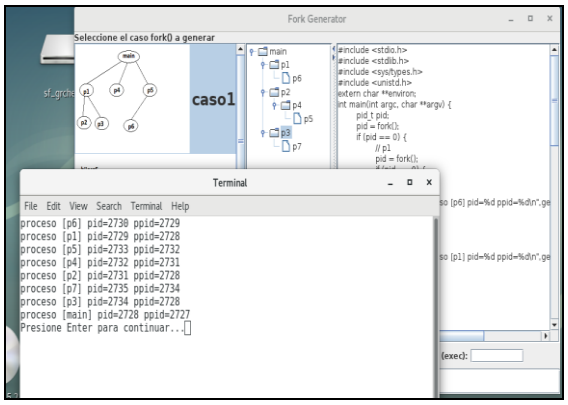

Figura 7. Fork Generator, Ejecución.

El software puede funcionar en cualquier plataforma en donde se instale Java Runtime Environment (J2RE)®[13], no obstante, para ejecutar el código generado requiere de una plataforma Unix® o Linux que disponga del compilador lenguaje C `gcc`[14] instalado y el comando `x-terminal-emulator`[15]. El software ha sido desarrollado y probado en plataforma Linux Debian 9 32 bits[16], se distribuye bajo licencia LGPL[17] y puede ser descargado desde http://www.grch.com.ar/docs/soft/forkgen/ .

4. Resultados

Se considera que el proyecto ha alcanzado el objetivo propuesto.

El software ha sido utilizado por los alumnos que este año están cursando la Asignatura Sistemas Operativos de la carrera Ingeniería en Sistemas de Información de la Universidad Tecnológica Nacional, Facultad Regional Delta. Se hizo énfasis en obtener comentarios por parte de los alumnos que habían tenido dificultades en el primer parcial en donde se evaluaron estos conceptos (y aún no estaba disponible esta herramienta) y ello nos alienta a utilizar la herramienta desde el comienzo del próximo ciclo lectivo, esperando que el uso de la misma permita un mejor rendimiento académico de los alumnos.

Se hicieron pruebas generando cada una de las combinaciones posibles que tiene el software, se han depurado todos los errores encontrados, el software está disponible para su uso productivo. Los tiempos de generación de código y ejecución son adecuados. El software exporta el código generado para su uso por fuera de la herramienta.

Se requiere contar con una estrategia pedagógica en la inserción del uso de esta herramienta, en el contexto del dictado de la asignatura. La herramienta no sólo tiene impacto en el proceso de enseñanza – aprendizaje de los alumnos sino también en la sistematización de ejercicios, su evaluación y corrección.

5. Discusión

Se espera que luego de la publicación y difusión del software se cuente con información por parte de usuarios que comiencen a usarlo. La interfase está en idioma español, su traducción a otros idiomas puede facilitar el acceso a una comunidad de usuarios más amplia.

Las pruebas se han realizado sobre plataforma Linux Debian 9 de 32 bits, utilizando openjdk-8[18], se espera que la comunidad de usuarios permita ampliar la prueba a otras plataformas. El software puede ejecutarse en Windows® aunque sólo para generar código, no es posible compilarlo y ejecutarlo debido a inexistencia de la llamada al sistema *fork* en Windows®.

Se ha generado un código compacto, teniendo en cuenta la complejidad de la aplicación, permite un trabajo interactivo fluido con el usuario y los requerimientos de hardware son idénticos al requerimiento de Java JRE® versión 8. Las pruebas de sucesivas ejecuciones no permiten apreciar una degradación en la performance del sistema.

La estrategia pedagógica de inserción de esta herramienta en la asignatura debe evitar caer en la mera memorización de algoritmos por parte de los alumnos, debe facilitar al alumno alcanzar la comprensión con respecto a la creación de procesos emparentados concurrentes.

6. Conclusión y Trabajo a Futuro

Se ha logrado concebir un software que puede ser una herramienta valiosa en el aprendizaje de la programación de procesos emparentados concurrentes, que suele presentar dificultad en los alumnos que cursan la asignatura Sistemas Operativos en la carrera de Ingeniería en Sistemas de Información.

La herramienta por sí sola no resuelve el problema de rendimiento académico de los alumnos, requiere de una estrategia pedagógica para su correcta inserción en la asignatura con el objetivo de mejorar los resultados obtenidos hasta el momento.

El software también puede utilizarse en aplicaciones comerciales, para la generación de código, en el desarrollo de este tipo de procesos y en la comparativa de rendimiento de procesos de este tipo, a partir de distintas estructuras jerárquicas.

Este trabajo abre nuevas líneas de investigación en cuanto a:

- diseño de una estrategia pedagógica para la inserción de esta herramienta en el contexto de las prácticas de la asignatura
- mejoras que puedan introducirse al software
- generación de código para otros lenguajes de programación
- ampliación de idiomas de la aplicación
- distribución en repositorios de aplicaciones de alcance global
- estudios cognitivos en cuanto a las estructuras jerárquicas de estos procesos y las dificultades que tienen los alumnos en su resolución
- cambios en la arquitectura de la aplicación: por ejemplo, si es posible su versión para dispositivos móviles; si es posible una versión orientada a internet o intranet

7. Agradecimientos

A los alumnos que actualmente cursan la Asignatura Sistemas Operativos, de Ingeniería en Sistemas de Información de la Universidad Tecnológica Nacional, Facultad Regional Delta, por su participación en el uso del software “Fork Generator”. A los alumnos que ya cursaron esta Asignatura, por compartir sus dificultades en este tema.

8. Referencias

[1]Tanenbaum Andrew S, Sistemas Operativos Modernos, 3ra Edición, Pearson Educación, México, 2009, ISBN 978-607-442-046-3, pp. 86-88
[2]Aho, Alfred V.; Hopcroft, John E.; Ullman, Jeffrey D.,Estructuras de datos y algoritmos. México: Addison Wesley, 1998, pp. 2-4.
[3] Stallings William, Sistemas Operativos, Pearson Educación, Madrid, 2005,ISBN 978-84-205-4462-5, pp. 402-408
[4] Deitel, Harvey M., An introduction to operating systems, Addison-Wesley, ISBN 0-201-14502-2. pp. 57-58
[5] Stevens Richard W., Rago Stephen A., Advanced Programming in the Unix® Environment, Third Edition, Addison-Wesley, 2013, ISBN 978-0-321-63773-4, pp. 533-588
[6] Silberschatz Abraham, Galvin Baer Peter, Gagne Greg, Operating System Concepts, Ninth Edition, Wiley, 2013, ISBN:978-1-118-06333-0, pp. 121-122
[7] Burgess, Mark, The Unix Programming Environment, Centre of Science and Technology, Faculty of Engineering, Oslo College, Edition 2.2, 2001, pp. 19-25
[8] Execvp, Linux Man Pages, disponible en https://www.systutorials.com/docs/linux/man/3-execvp/
[9] Lenguaje de Programación Java, Oracle Corporation, disponible en https://www.java.com/es/download/faq/whatis_java.xml
[10] O'Conner, John, Using the Swing Application Framework (JSR 296), Oracle Corporation, 2007, disponible en https://www.oracle.com/technetwork/articles/java/swingappfr-136951.html
[11] Kernighan Brian W., Ritchie Dennis M, El Lenguaje de Programación C, Segunda Edición, Pearson Educación, ISBN 968-880-205-0
[12] Java Tutorials, Using Swing Components, Oracle Corporation, disponible en https://docs.oracle.com/javase/tutorial/uiswing/components/tree.html
[13] Java Runtime Environment, Oracle Corporation, disponible en http://www.oracle.com/technetwork/java/javase/overview/index.html
[14] GCC, GNU Compiler Collection, Free Software Foundation, disponible en https://gcc.gnu.org/
[15] x-terminal-emulator, gnome-terminal, manual page disponible en https://helpmanual.io/man1/x-terminal-emulator/
[16] Linux Debian, The Universal Operating System, disponible en https://www.debian.org/
[17] Free Software Foundation, GNU Lesser General Public License, Version 3, June 2007, disponible en https://www.gnu.org/licenses/lgpl-3.0.en.html
[18] OpenJDK, Oracle Corporation, 2018, disponible en http://openjdk.java.net/

Datos de Contacto:

Cherencio, Guillermo Ruben. Universidad Tecnológica Nacional, Facultad Regional Delta. San Martin 1171, CP 2804, Campana, Provincia de Buenos Aires, República Argentina. E-mail: grchere@yahoo.com ; Romero, Juan Carlos E-mail: juancarlosjromer@gmail.com ; Perello, Mario Gerardo E-mail: mperello04@yahoo.com.ar
.