PROYECTO

Universidad Católica Andrés Bello

Facultad de Ingeniería

Escuela de Informática

Materia: Sistemas de Operación

Profesor: Richard Duarte

RESUMEN. El presente documento es una definición general de los problemas a desarrollar para el actual semestre de *Sistemas Operativos*, los cuales deben ser desarrollados en grupo y entregados según las especificaciones y pasos listados en el presente documento.

1. REQUERIMIENTOS

A continuación, se presenta una lista con los requerimientos del proyecto:

1. Escribir un ***makefile*** (Linux) el cual construya todos los programas necesarios para resolver los problemas que se exponen a continuación. **(2 ptos)**

2. Escribir un programa que determine cuántos procesos activos a la vez puede tener un usuario. **(3 ptos)**

3. Escribir un programa denominado timeprog. Este programa recibirá como parámetros un programa (prog) y sus argumentos, para determinar cuánto tiempo demora la ejecución de dicho programa (prog). La sintaxis para timeprog es la siguiente:

timeprog <prog><[arg1, arg2, ..., arg9]>

Los parámetros son opcionales y dependerán del programa prog a ser ejecutado por timeprog. La salida de timeprog deberá ser el número de segundos requeridos durante la ejecución de prog. (3 ptos)

4. Escribir una versión propia del programa para copiar archivos de Unix, cp.

Denominar esta nueva versión ucp. La sintaxis para ucp es la siguiente:

ucp bufsize file1 file2

En donde **bufsize** es un entero que determina el tamaño del ***buffer*** para la lectura de la data desde ***file1*** y la escritura de la data en ***file2***. El programa deberá aceptar valores para ***bufsize*** dentro del rango 1 a 16384. Utilizar timeprog de la pregunta anterior para probar ucp copiando un archivo grande (640K o mayor). Los valores de prueba para bufsize deberán ser: 1, 32, 8192 y 16384. ¿Cuáles son los resultados? (5 ptos)

5. Escribir un programa en donde un padre crea dos procesos hijos, ***C1*** y ***C2*** y crea un pipe para que ***C1*** se comunique con ***C2***. Programar ***C1*** para que reciba una señal **SIGALRM** cada 2 segundos. Cada vez que ***C1*** reciba una señal **SIGALRM** envía el mensaje recibido a ***C2***, el cual lo imprime por pantalla al recibirlo. ***C1*** deberá terminar una vez que haya recibido 10 señales **SIGALRM** y enviado 10 mensajes a ***C2***. El proceso ***C2*** deberá terminar una vez que haya recibido todos los mensajes y los haya impreso por pantalla. Terminar el proceso padre una vez que ambos hijos hayan terminado. (5 ptos)

6. Realizar dos (2) versiones para el cálculo de "**Sucesión de Fibonacci"**. La primera versión es *secuencial* la cuál deberá sumar un millón de números de **FIBONACCI** y generar de forma aleatoria entre 0 y 19 y a ese número le calculamos su **FIBONACCI**, generar ese millón de números hacer la suma e imprimirlo por pantalla (usar las siguientes librerías: stdio.h, math.h, stdlib.h y time.h). La segunda versión deberá ser en paralelo utilizando "HILOS" e imprimirlo por pantalla, el objetivo de este ejercicio es observar cual versión correr más rápido (usar función "pthread"). (10 ptos)

2. DESARROLLO

1. El proyecto deberá ser desarrollado utilizando las herramientas de UNIX/Linux para el desarrollo. Básicamente un editor de texto, un compilador y make como mínimo.

2. El lenguaje de programación deberá ser **C** estándar, no se permitirán frameworks.

3. El proyecto deberá ser desarrollado en equipos de 4 o máximo 5 integrantes.

4. Es responsabilidad de cada grupo dividir el trabajo entre los integrantes, pero todos los integrantes deberán trabajar en el proyecto.

5. El código a entregar deberá estar totalmente documentado y ser lo más legible posible.

6. No se permitirá el uso de ambientes de desarrollo (IDE).

3. REGLAS Y EVALUACIÓN

1. La colaboración entre los grupos de trabajo es fomentada pero no la copia.

2. Todo proyecto que se consiga con un código sintácticamente o semánticamente parecido tendrá una nota de 0 puntos.

3. Cada grupo será evaluado por los problemas desarrollados.

4. La evaluación será en grupo, pero también individual.

5. Cada grupo será responsable de la entrega de su proyecto.

6. El proyecto deberá ser entregado antes de la fecha y hora límite, de lo contrario no será evaluado.

8. Todos los integrantes de los grupos deberán intentar realizar la misma cantidad de trabajo.