**INSTITUTO TECNOLÓGICO AUTÓNOMO DE MÉXICO**

**PRÁCTICAS de SISTEMAS OPERATIVOS**

# PRÁCTICA #7

# <Threads, programación concurrente>

**Grupo**

**<G5>**

**Integrantes**

**<Rebeca Baños García – 157655>**

**<Víctor Hugo Flores Pineda– 155990>**

**<Humberto Martínez Barrón y Robles – 166056>**

**Fecha (s) de elaboración de la práctica**

**<3 de junio del 2020>**

# Práctica 07
# Threads, programación concurrente.

INTRODUCCIÓN: Los hilos, o threads, son los flujos de ejecución del CPU en un programa, es decir, cada hilo que contiene un programa se encarga de determinar el orden de ejecución secuencia que seguirá el flujo del programa al que pertenece el hilo. Los threads corren dentro del contexto del proceso pesado y toma ventaja de los recursos y del ambiente. Así que se podría decir que los hilos son la unidad de planificación.
Existe un mecanismo para tener dos o más hilos o threads dentro de un solo proceso pesado, esto se le conoce como Multithread progrmming. En este caso, hay un hilo principal, que es donde inicia la ejecución de proceso.
Los hilos utiliza un program counter, un conjunto de registros y un stack, esto para tener el orden de ejecución correcto de cada proceso en el flujo. Dentro del proceso, los threads comparten el área de código y de datos, así como los recursos del sistema operativo.

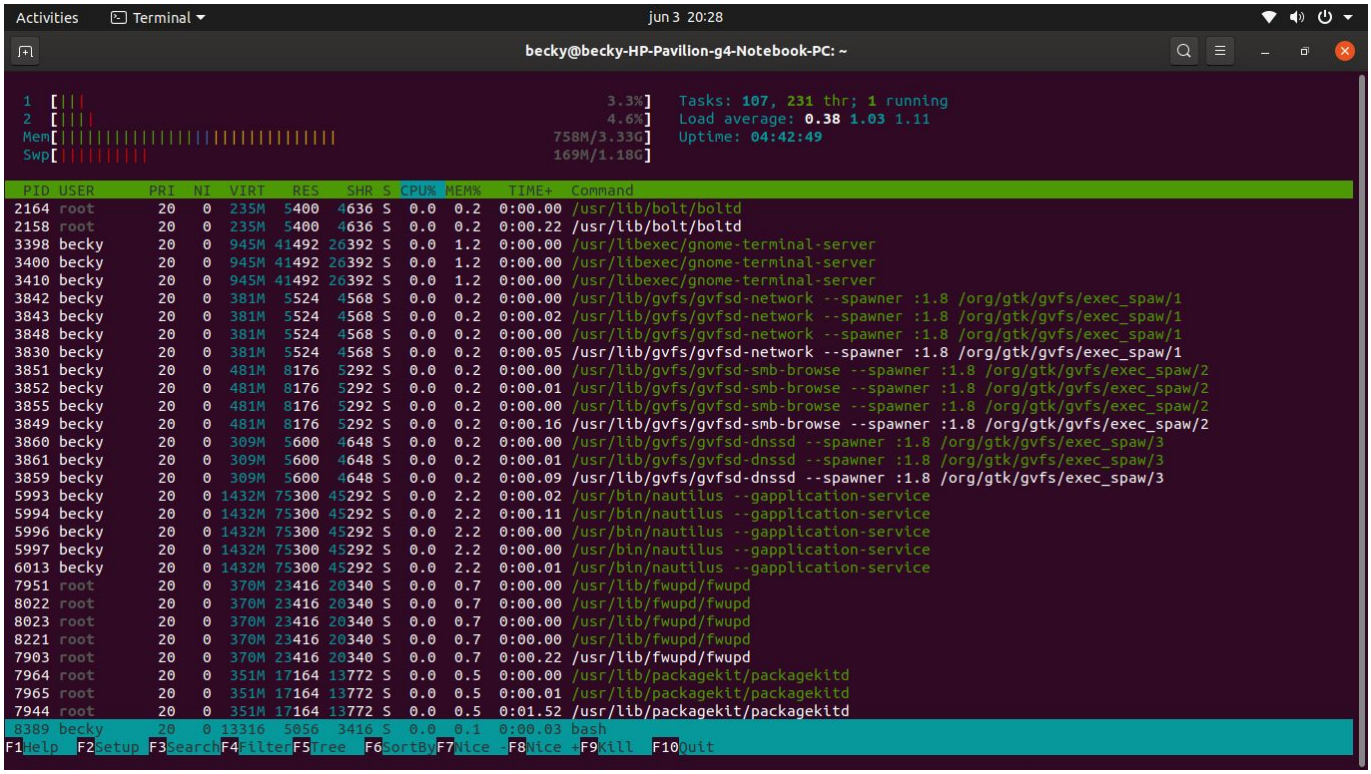
1- Retome la aplicación concurrente (Práctica 06), de la generación de estados de cuenta, resolviéndola ahora con *threads* en lugar de *procesos pesados*.

   Usted deberá conservar toda la funcionalidad pedida anteriormente, que usted ya la llevó a cabo en la solución de la Práctica 06, por lo que tendrá que transportar esta funcionalidad a esta programación multithread.

   Ahora usted tendrá en *EdoCtaClientesC.java* la clase con el programa principal mientras que *MovimientosClienteC.java* será la clase Thread derivada. Así durante la ejecución tendremos el Thread "padre" que inicia en *main(...)* en *EdoCtaClientesC*, así como los Threads "hijos" *MovimientosClienteC* que elaborarán los estados de cuenta.

   Además, recolecte tres tiempos de ejecución tanto de la aplicación concurrente con procesos pesados (Práctica 06) como otros tres de la actual Práctica 07.
   Tome muestras tanto del padre como de algún hijo.
   Práctica 06: 1595 ms, 1341 ms, 1448 ms.
   Práctica 07: 77 ms , 48 ms, 43 ms.

   Con *Process Explorer* capture un "pantallazo" de la ejecución de esta aplicación.




Ahora, en esta práctica 07, ¿Cuántos y cuáles son los programas ordinarios que conforman el programa concurrente? Los programas ordinarios son EdoCtaClientes.class y MovimientosClientes.class
En esta misma práctica 07 ¿Cuántos procesos ligeros (threads) fueron ejecutados? Fueron 12 procesos ligeros, uno por cada cliente que se ejecutó y ¿Cuántos procesos tradicionales o pesados? No se ejecutaron procesos pesados ya que fueron sustituidos por threads

¿De los resultados recolectados como justifica que este programa concurrente con *threads* es más barato en recursos y más rápido? Porque el tiempo de ejecución fue muy rápido a comparación de cuando se hizo en la práctica 6 y solamente despliega los errores encontrados.

CONCLUSIONES: Por los resultados obtenidos, pudimos notar que utilizar threads para diferentes programas ayudan a que estos se ejecuten más rápido, ya que el orden de flujo no cambia. Otro aspecto importante es que al correrlo en la terminal, solamente despliega los mensajes de error y no todo lo que se está ejecutando, esto es porque lo que se ejecuta siempre es lo mismo en cuánto al orden y solamente cambia el nombre de cada cliente que se está ejecutando.
Fue una práctica interesate para poder comparar la diferencia entre los procesos pesados u ordinarios y el uso de threads ya que dependiendo la tarea que tengamos que realizar, utilizaremos la herramienta más adecuada.