Tareas Abc

La solución consiste de :

i) Un creador de tareas que simula la entrada de tareas de usuario
ii) Un coordinador principal
iii) Un coordinador para cada tipo de tarea
iv) Las funciones que realizan las tareas
v) Un programa main que inicializa todo y lo pone en ejecución
vi) Y 4 pipes de comunicación uno por coordinador

En main se abren los pipes, se crean 3 hijos del proceso actual a los cuales se les asigna un coordinador diferente a cada uno, y luego se manda al padre a ejecutar el coordinador principal

El coordinador principal, simula la creación de tareas(con fines de testeo pero en realidad esperaría hasta que le llegue una cantidad de tareas enviadas por los usuarios), envía las tareas por los pipes correspondientes y espera a que todos los coordinadores le avisen que terminaron con sus tareas para recibir más.

El coordinador A,B y C funcionan de la misma manera, reciben la cantidad de tareas que deben realizar (de su tipo) y crean una cantidad de hilos equivalentes, luego hacen que esos hilos ejecuten la función correspondiente enviando los datos relevantes para su ejecución. A medida que las tareas van terminando envían por el pipe del coordinador principal un aviso de que una de las tareas que debían realizar ya esta terminada, después de que todas sus tareas finalizan se queda esperando a que el coordinador principal le envíe nuevas tareas por el pipe.

Mini Shell

Objetivo: Implementación de un mini Shell con comando limitados.

Opciones de diseño:

Se eligió implementar el miniShell como un programa que se ejecuta en consola, pero que no funciona como tal. Por ejemplo al crear un directorio, luego de escribir el comando de creación de directorio, se mostrará por consola un mensaje que le indica al usuario que escriba el nombre del directorio a crear.

En el momento en el que se introduce un comando, es posible escribir otras cadenas de caracteres, pero estas serán leídas luego de terminar de ejecutarse el comando introducido inicialmente. Luego dichas cadenas serán tomadas como nuevos comandos.

Cuando se ejecuta un comando, se indicará que introduzca datos, en todos los comandos implementados menos en modPer, sólo será válida la primera cadena de caracteres, las consiguientes serán ignoradas. En modPer sólo serán válidas las dos primeras cadenas de caracteres, las consiguientes serán ignoradas.

Se optó por no implementar el comando “cd”, por lo que los comandos se realizarán en el directorio donde se encuentre el programa.

Al crear un directorio, sus permisos serán por defecto “0777”.

Si un directorio no está vacío no se podrá eliminar y se mostrará un mensaje de error por consola.

Al momento de modificar los permisos de un archivo o directorio, el modo debe introducirse como un número octal de cuatro cifras, serán incorrectas las demás formas de escribir modos y se mostrará por consola un mensaje de error.

Al introducir un comando erróneo o no implementado se mostrará un mensaje por consola en el que se indica como tal y que existe el comando “help”, el cual mostrará por consola los comandos posibles.

Se implementó el miniShell de tal manera que se ejecutará hasta que se fuerce el cierre o se introduzca el comando “exit”, el cual terminará su ejecución.

Los comandos se implementaron sensibles a mayúscula.

Comida Rápida

Hilos:
-50 clientes que están esperando desde el instante 0
-3 cocineros cocinando desde el instante 0
-1 camarero listo para atender desde el instante 0
-1 limpiador listo para limpiar desde el instante 0

Semáforos:
-Un semáforo simula los lugares vacíos en el espacio para platos de la cocina iniciado en 10
-Un semáforo contraparte del semáforo anterior que simula los lugares llenos iniciado en 0
-Un semáforo que representa las mesas limpias que iniciado en 30
-Un semáforo contraparte del anterior que simula las mesas sucias iniciado en 0
-Un semáforo para que los camareros sepan que hay clientes sin atender
-Un semáforo para que los clientes esperen a ser atendidos

Comportamiento de la solución:
0: los cocineros empiezan a cocinar hasta que llenen la cocina
1:los limpiadores están esperando que se ensucie una mesa
2:los camareros están esperando que algún cliente se siente
3:los clientes empiezan a llegar y ocupan la primer mesa limpia que encuentren
5: el camarero comienza a atender a los clientes que están en las mesas si tiene comida en la cocina
6: el primer cocinero que vea que hay lugar en la cocina se pone a cocinar y lo llena
7: los clientes que no consiguieron mesa están esperando mesas limpias
8: los clientes atendidos se van y dejan la mesa sucia
9: el limpiador ve la mesa sucia y la limpia

10:el primer cliente que la encuentra la toma

el ciclo 5,6,8,9,10 se repite hasta que no quedan clientes esperando.

El problema que yo veo con esta solución es que quedan diez platos de comida sin usar cada vez que termina el día del restaurante. y los semáforos no quedan correctamente actualizados para comenzar una nueva ejecución desde 0.

Lectura : Android Operating System Architecture

a) Describa los componentes de la arquitectura:
    i) Kernel Layer.
        1) Binder: Maneja la comunicación entre procesos (IPC) más eficientemente que el kernel de linux. Las aplicaciones y servicios pueden ejecutarse en diferentes procesos pero deben comunicarse y compartir información.
        2) Alarm Driver: El kernel de android realiza ciertas operaciones y las planifica apropiadamente para despertar una aplicación cuando un evento provoca un Trigger.
        3) Power Management: Además del administrador de batería de linux, android trae su propio administrador de batería que se adapta a las necesidad de un teléfono inteligente.
        4) Low memory killer: Cuando un dispositivo se queda sin espacio en memoria, elige un proceso y lo mata; el proceso es seleccionado utilizando una política, elegida por el propio usuario.
        5) Kernel Debugger: Para hacer un seguimiento de los cambios, funcionen o no, android es proveído de un Kernel Debugger.
        6) Logger: Otra parte importante añadida al Kernel es el logger, que registra los mensajes del sistema con el propósito de solucionar problemas. Están incluidos logger buffer, event logger buffer, radio logger buffer y system buffer.
        7) Ashmem : Es la abreviación de memoria compartida de android, añadida al kernel para facilitar el intercambio de memoria y provee mejor soporte a dispositivos con poca memoria porque puede descartar unidades de memoria compartida bajo presión.
    ii) Native Libraries layer.
        1) Libc: Android implementa su propia versión especial de bionic, libc, que es una pequeña en tamaño comparada con las librerías de GNU (glicb), que son demasiado grandes y complicadas para teléfonos celulares.
        2) SQLite: SQLite es una base de datos relacional usada en android para almacenamiento de datos que está disponible para todas las aplicaciones, para almacenar o recuperar datos conforme a sus necesidades.
        3) Media Framework: Provee grabación y reproducción de numerosos formatos de audio, video e imágenes que incluyen MPEG4, MP3, AME, H.264, AAC, PNG y JPG.

4) Surfaceflinger: Provee a todo el sistema de un manejador de composición de superficies, compone los distintos buffers de información que debe salir por el display en un solo buffer y los envía al Hardware de abstracción de capas(HAL).
5) WebKit: es un motor de navegación que es usado para mostrar contenido HTML.
6) La segunda parte de este nivel es el “Tiempo de ejecución” de Android, que consiste de la máquina virtual Dalvik(DVM) y librerías esenciales de java. Las aplicaciones de Android y los framework principales están escritos en el lenguaje de programación Java. La capacidad de esto ha sido posible gracias a la máquina virtual Java(JVM), Android utiliza su propia DVM que está diseñada para pequeños sistemas, estos sistemas usualmente proveen poco ram y una CPU lenta. Como la JVM, Dalvik tiene su propio formato de byte que se ajusta a las necesidades de los dispositivos Android elegidos. Este código de byte está más comprimido que el típico código de byte de Java. Sus archivos de ejecución en el formato del ejecutable Dalvik (.dex) está optimizado para minimizar el uso de la memoria principal. DVM provee a las aplicaciones portabilidad y un tiempo de ejecución consistente y permite correr en su propio procesos, con su propia instancia de DVM.

iii) Application Framework layer.

1) Una actividad representa una única pantalla con una interfaz de usuario.En aplicaciones, las actividades trabajan en conjunto para conseguir una experiencia cohesiva. En múltiples actividades de una aplicación, típicamente, una actividad es elegida como “la principal” que es presentada al usuario al iniciar la aplicación por primera vez. Para realizar diferentes acciones cada actividad puede comenzar otra actividad. Cuando una nueva actividad comienza, la anterior es detenida y se guarda su estado en una pila para poder utilizarla en otro momento. Hay un manejador de actividades que maneja el ciclo de vida de las aplicaciones.
2) Un servicio es un componente de Android que se ejecuta en segundo plano y no provee interfaz de usuario. Los servicios ejecutan operaciones de larga duración y trabajan para procesos remotos.
3) Content Providers: Datos compartidos entre aplicaciones son compartidos por el proveedor de contenido. Este maneja cómo se accede a los datos desde otras aplicaciones.
4) Package Manager: Android usa una extensión especial para paquetes llamada APK(Android Package)
5) Window Manager: Dibuja diferentes ventanas en la pantalla para interactuar con ellas usando el buffer del Surfaceflinger.
6) Hardware Services: Interactúa con el nivel abstracto de hardware para acceder a dispositivos, el acceso directo a estos no es permitido, hay un grupo de gestores involucrados en el acceso a dispositivos.
7) Telephone Service: Relacionado con las llamadas telefónicas y el envío de mensajes.

8) Location Service: Es usado para la gestión de localización. GPS o antenas de telefonía móvil son usadas para conseguir la localización del dispositivo.

iv) Application layer.

En android la capa de aplicación es la capa superior. Estos comprenden las aplicaciones nativas al igual que las pre-instaladas con cada dispositivos, como: Phone dialer, SMS Client, Web Browser contact Manager etc.

En promedio un usuario de un dispositivo android interactúa mayormente con esta capa para funciones básicas como llamadas telefónicas, enviar mensajes, capturar fotografías, navegar en la web, reproducir videos y audios. Un desarrollador puede escribir su propia aplicación y reemplazar aplicaciones existentes. También hay disponibles aplicaciones fuera de la Google play store . Los usuarios pueden fácilmente descargar y usar estas aplicación conforme a sus necesidades.

b) Identifique elementos que son representativos para este tipo de sistemas operativos.

i) Máquina virtual Dalvik y cigoto.
ii) Navegador integrado.
iii) SQLite.
iv) GSM Technology.
v) Media Support.
vi) Desarrollada utilizando programación java.
vii) Boot Loader
1) Cuando se presiona el botón de encendido, el código Boot ROM empieza a ejecutar desde una locación pre definida en la ROM Esto carga el Boot loader en la Ram y comienza a ejecutar. No es parte del sistema operativo pero me parece importante destacarlo.
viii) Activity Manager
ix) Launcher (Home)
1) Es lo último en ejecutar y mostrarse en pantalla y es lo que permite la mayoría de interacciones con el usuario.
x) La seguridad es muy difícil de asegurar.

c) Realice un comentario general del artículo (al menos 500 palabras).

El artículo provee un resumen sobre el Sistema operativo Android desarrollado por Google, los datos más destacables son que, a pesar de correr sobre un Kernel de Linux, está ampliamente modificado para ser más funcional a los celulares o dispositivos con sus limitados recursos, apuntando a reducir el coste de recursos de algunos mecanismos, y mejorar la velocidad en otros tantos. Otro punto que destaca el artículo es el tema de la seguridad y como los mecanismos de seguridad de Android, mas haya de haber mejorado mucho en los últimos años siguen sin ser igual de efectivos que los de otros Sistemas operativos como pueden ser Apple iOS, Blackberry OS, etc, esto se debe principalmente a su condición de ser de código abierto, aunque esto lo hace muy atractivo para desarrolladores de aplicaciones, al descubrir una falla o agujero de seguridad se debe avisar esencialmente a todo el mundo y si un parche que solucione estos problemas no sale rápidamente, este es fácilmente explotable ya que todo el mundo lo conoce. Al ser un software de código

abierto, permite implementar distribuciones del sistema operativo independiente a aquellas pretendidas por los creadores de dispositivos, esto agrega fallas que pueden ser aprovechadas por código malicioso.
Lo otro que lo hace más vulnerable, es el hecho de usar una arquitectura monolítica heredada del Kernel de Linux, implicando que todos los componentes están acoplados y funcionan como una sola pieza, y muchos creadores de dispositivos Android usan sus drivers personalizados o de la empresa para su hardware, generando la posibilidad de meter fallas de seguridad, y al no facilitar este código a la comunidad de Linux esta queda sin testear por la misma.
La capacidad de “rootear” los dispositivos también es un factor por el que la seguridad se ve comprometida.Cuando alguien “rootea” su celular o dispositivo android, este dispositivo ya no es seguro, debido a que el procesos involucrado en rootear el dispositivo rompe la barrera de integridad del Kernel. Un sistema rooteado no puede estar seguro sobre el contenido de su Kernel modificado ya que este podría deshabilitar las medidas de seguridad del sistema para contener algún tipo de Malware, como podría ser un guardador de contraseñas e información personal.
En resumen android es un sistema operativo de código abierto e implementado en su mayoría con Java, lo que le permite una alta compatibilidad al correr sobre la máquina virtual de Java, tiene su propia versión reducida llamada Dalvik, lo que otorga a desarrolladores una gran libertad en comparación con sus competidores más conocidos, esto lo ha convertido en uno de los más populares, si no en el más popular, tiene un buen rendimiento pues todos sus mecanismo están diseñados para funcionar en móviles o dispositivos con recursos limitados pero cuentan con varias debilidades en el área de la seguridad, dificultad de homogeneizar las aplicaciones para diferentes dispositivos, ya que estos no siguen los mismos estándares que sus competidores más directos, y es enviado de fábrica con varios middlewares que puede que el usuario nunca utilice.