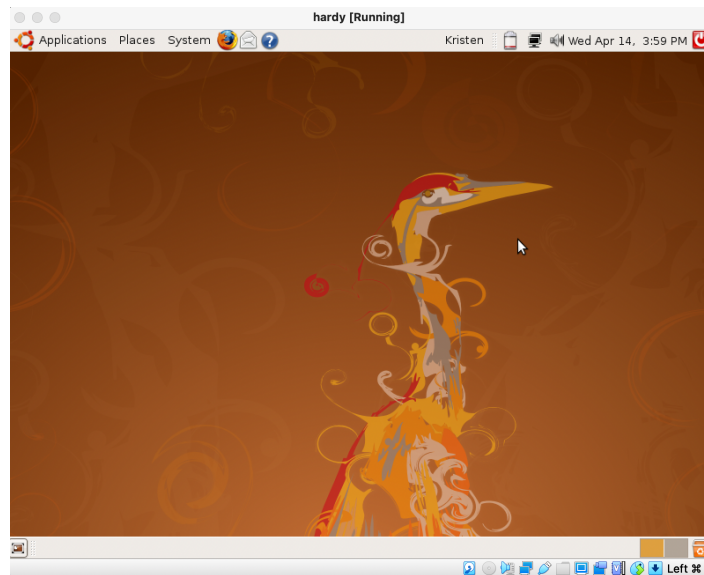


Kristen Brandt
171482

Lab 5



```
kristen@kristen-laptop: ~/Documents/tasks
File Edit View Terminal Tabs Help
GNU nano 2.0.7 File: pre_casio.txt.

pid[2]
deadline[1500000000]
Before sched_setscheduler:priority 0
After sched_setscheduler:priority 0

Task(2) has just started
Job(2,1) starts
Job(2,1) ends (1.020000)
Job(2,2) starts
Job(2,2) ends (1.090000)
Job(2,3) starts
Job(2,3) ends (1.070000)

Task(2) has finished

pid[3]
deadline[1400000000]
Before sched_setscheduler:priority 0
[ Read 63 lines ]
^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell
```

```
kristen@kristen-laptop:/home$ sudo mkdir scheduler_div
kristen@kristen-laptop:/home$ sudo mkdir scheduler
kristen@kristen-laptop:/home$ ls
kristen scheduler scheduler_div
kristen@kristen-laptop:/home$
```

```
kristen@kristen-laptop:/home$ sudo chown -R kristen scheduler{,_dev}
kristen@kristen-laptop:/home$ ls
kristen scheduler scheduler_dev
kristen@kristen-laptop:/home$ ls -Al
total 12
drwxr-xr-x 27 kristen kristen 4096 2021-04-17 22:28 kristen
drwxr-xr-x  2 kristen root    4096 2021-04-17 22:42 scheduler
drwxr-xr-x  2 kristen root    4096 2021-04-17 22:43 scheduler_dev
kristen@kristen-laptop:/home$
```

- Funcionamiento y sintaxis de uso de structs.

Los structs son un tipo de dato que el usuario define que permite la combinación de elementos de datos de distintos tipos. Para definir la estructura de se usa la instrucción “struct”. La sintaxis de los structs es de esta manera:

```
struct [structure tag] {

    member definition;

    member definition;

    ...

    member definition;

} [one or more structure variables];
```

https://www.tutorialspoint.com/cprogramming/c_structures.htm

- Propósito y directivas del preprocesador.

El preprocesador en C no es el compilador sino que es una parte separada dentro del proceso de compilación. El preprocesador es una herramienta que le indica al compilador que haga el preprocesamiento antes de que se compile. Los comandos de los preprocesadores empiezan con un hashtag (#) pegado a la izquierda del documento. Las directivas del preprocesador son:

1. #define
2. #include
3. #undef
4. #ifdef
5. #ifndef
6. #if
7. #else
8. #elif

9. #endif
10. #error
11. #pragma

- Diferencia entre * y & en el manejo de referencias a memoria (punteros).

& es la dirección del operador/variable mientras que * es el que nos da el valor de las referencias a memoria.

- Propósito y modo de uso de APT y dpkg.

Apt-get utiliza dpkg para poder realizar las instalaciones o eliminaciones del paquete que se desea. dpkg también sirve para darnos información específica de un programa. Apt puede descargar e instalar paquetes que necesitan que otros estén previamente instalados, dpkg no puede instalar paquetes si necesita que otros estén instalados previamente.

- ¿Cuál es el propósito de los archivos sched.h modificados?

El propósito de los archivos sched.h que se modificaron es que se define que calendarizador se va a utilizar. Estos archivos también nos ayudan a implementar las estructuras hechas en sched_param.

- ¿Cuál es el propósito de la definición incluida y las definiciones existentes en el archivo?

El propósito de la definición incluida es que si esa definición ya está escrita no se vuelva a definir en otro lugar. El propósito de esto es evitar el conflicto que se puede generar.

- ¿Qué es una task en Linux?

Un task en Linux es algo utilizado por el kernel que nos ayuda a referirse a una unidad de ejecución. El task puede compartir recursos del sistema con otras tareas del sistema. Depende del nivel de intercambio del task este puede considerarse como un hilo o un proceso.

- ¿Cuál es el propósito de task_struct y cuál es su análogo en Windows?

El propósito de task_struct es contener toda la información necesaria acerca de un proceso. En windows su análogo es struct thread_info.

- ¿Qué información contiene sched_param?

Sched_param se utiliza cuando se obtienen o establecen los parámetros de programación para un hilo o proceso. La información que esta tiene son los parámetros de calendarización.

- ¿Para qué sirve la función `rt_policy` y para qué sirve la llamada `unlikely` en ella?

La función `rt_policy` sirve para decidir si una política de programación determinada pertenece a la clase de tiempo real o no. La llamada `unlikely` dentro de esta función sirve para crear condiciones para los posibles procesos.

- ¿Qué tipo de tareas calendariza la política EDF, en vista del método modificado?

EDF o Earliest Deadline First utiliza un esquema en el que el proceso más cercano a su fecha límite es el siguiente proceso que será ejecutado. El EDF se utiliza para el manejo de tareas en sistemas que necesitan cumplir con requisitos de tiempo.

- Describa la precedencia de prioridades para las políticas EDF, RT y CFS, de acuerdo con los cambios realizados hasta ahora.

La precedencia de prioridades para las políticas EDF, RT y CFS es:

EDF: Algoritmo que utiliza un esquema en el que el proceso más cercano a su fecha límite es el siguiente proceso que será ejecutado.

RT: Microkernel que ejecuta el sistema operativo de forma preemptive.

CFS: Es un planificador de procesos. Este es el encargado de asignar recursos de la computadora para la ejecución de los procesos de tal manera que se maximice la utilización del CPU.

- Explique el contenido de la estructura `casio_task`.

La estructura `casio_task` representa la entidad de un proceso en tiempo real de casio. Esta estructura está encargada de decir cómo funcionan los nodos de red black tree, la raíz y la calendarización de las tareas. _____

- Explique el propósito y contenido de la estructura `casio_rq`.

El propósito de la estructura `casio_rq` es administrar todas las tareas de casio asignadas a un procesador. `Casio_rq` es el que inicializa el red black tree.

- ¿Qué es y para qué sirve el tipo `atomic_t`? Describa brevemente los conceptos de operaciones RMW (*read-modify-write*) y mapeo de dispositivos en memoria (MMIO).

`Atomic_t` sirve se define como un entero con signo. La operación RMW asegura que solo se modifiquen los bits específicos en un registro del sistema que se desea cambiar. MMIO es más eficiente que RMW ya que solo se cargan las regiones del archivo a las que realmente accede un programa.

```
typedef struct { int counter; } atomic_t;
```

<https://www.kernel.org/doc/html/v4.16/core-api/refcount-vs-atomic.html>

- ¿Qué indica el campo .next de esta estructura?

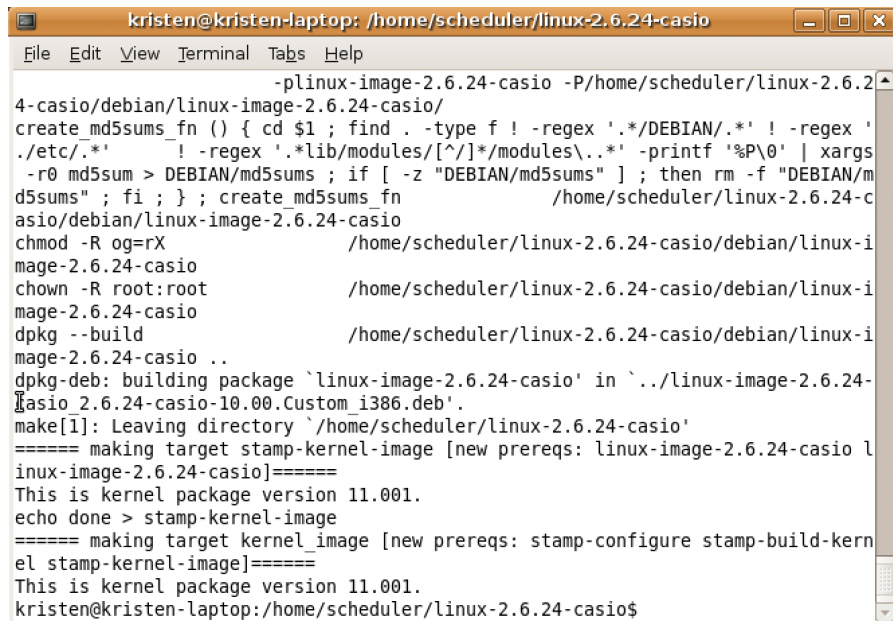
El campo .next indica que se realice una llamada al siguiente task. Esta llamada al task revisa la dirección de referencia de memoria de la última tarea.

- Tomando en cuenta las funciones para manejo de lista y red-black tree de casio_tasks, explique el ciclo de vida de una casio_task desde el momento en el que se le asigna esta clase de calendarización mediante sched_setscheduler. El objetivo es que indique el orden y los escenarios en los que se ejecutan estas funciones, así como las estructuras de datos por las que pasa. ¿Por qué se guardan las casio_tasks en un red-black tree y en una lista encadenada?

Los casio_tasks se guardan en un red-black tree porque así puede encontrar el siguiente task que tiene los criterios necesarios. Los casio_tasks se guardan en una lista encadenada porque esto lo necesita el modelo red-black tree.

- ¿Cuándo preemptea una casio_task a la task actualmente en ejecución?

Se preemptea una casio_task a la task actualmente en ejecución cuando el deadline del task siguiente es más cercano/menor al que actualmente se está ejecutando. _____



```
kristen@kristen-laptop: /home/scheduler/linux-2.6.24-casio
File Edit View Terminal Tabs Help
-plineux-image-2.6.24-casio -P/home/scheduler/linux-2.6.24-casio
4-casio/debian/linux-image-2.6.24-casio/
create_md5sums_fn () { cd $1 ; find . -type f ! -regex '.*DEBIAN/.*' ! -regex '
./etc/.*' ! -regex '.*lib/modules/[^/]*/modules\.*' -printf '%P\0' | xargs
-r0 md5sum > DEBIAN/md5sums ; if [ -z "DEBIAN/md5sums" ] ; then rm -f "DEBIAN/m
d5sums" ; fi ; } ; create_md5sums_fn /home/scheduler/linux-2.6.24-c
asio/debian/linux-image-2.6.24-casio
chmod -R og=rX /home/scheduler/linux-2.6.24-casio/debian/linux-i
mage-2.6.24-casio
chown -R root:root /home/scheduler/linux-2.6.24-casio/debian/linux-i
mage-2.6.24-casio
dpkg --build /home/scheduler/linux-2.6.24-casio/debian/linux-i
mage-2.6.24-casio ..
dpkg-deb: building package `linux-image-2.6.24-casio' in `./linux-image-2.6.24-
casio_2.6.24-casio-10.00.Custom_i386.deb'.
make[1]: Leaving directory `/home/scheduler/linux-2.6.24-casio'
===== making target stamp-kernel-image [new prereqs: linux-image-2.6.24-casio l
inux-image-2.6.24-casio]=====
This is kernel package version 11.001.
echo done > stamp-kernel-image
===== making target kernel_image [new prereqs: stamp-configure stamp-build-kern
el stamp-kernel-image]=====
This is kernel package version 11.001.
kristen@kristen-laptop:/home/scheduler/linux-2.6.24-casio$
```

```
kristen@kristen-laptop: /home/scheduler
File Edit View Terminal Tabs Help
[sudo] password for kristen:
(Reading database ... 99501 files and directories currently installed.)
Unpacking linux-image-2.6.24-casio (from linux-image-2.6.24-casio_2.6.24-casio-1
0.00.Custom_i386.deb) ...
Done.
Setting up linux-image-2.6.24-casio (2.6.24-casio-10.00.Custom) ...
Running depmod.
Finding valid ramdisk creators.
Using mkinitramfs-kpkg to build the ramdisk.
Running postinst hook script update-grub.
Searching for GRUB installation directory ... found: /boot/grub
Searching for default file ... found: /boot/grub/default
Testing for an existing GRUB menu.lst file ... found: /boot/grub/menu.lst
Searching for splash image ... none found, skipping ...
Found kernel: /boot/vmlinuz-2.6.24-casio
Found kernel: /boot/vmlinuz-2.6.24-26-generic
Found kernel: /boot/memtest86+.bin
Replacing config file /var/run/grub/menu.lst with new version
Updating /boot/grub/menu.lst ... done

Examining /etc/kernel/postinst.d.
run-parts: executing /etc/kernel/postinst.d/vboxadd

kristen@kristen-laptop:/home/scheduler$
```

