Notas para la práctica 1

Sistemas Operativos 2
Licenciatura en Ciencias de la Computación
2023

1. Introducción

1.1. ¿Qué es Nachos?

Nachos (siglas del inglés "Not Another Completely Heuristic Operating System") es un sistema operativo educativo para estudiantes de cursos de Sistemas Operativos en carreras de grado. Escrito originalmente en C++ para la arquitectura MIPS, Nachos se ejecuta como un proceso de usuario en el sistema operativo anfitrión. Un simulador de MIPS ejecuta el código para cualquier programa de usuario que se ejecute sobre el sistema operativo Nachos.

Fue desarrollado originalmente en la Universidad de California en Berkeley por Wayne A. Christopher, Steven J. Procter y Thomas E. Anderson entre 1991 y 1992, y lo usan numerosos centros de enseñanza. La versión que usamos en esta materia incorpora unos agregados hechos en 2007 por José Miguel Santos Espino, de la Universidad de Las Palmas de Gran Canaria. Desde 2015, además, los docentes de esta materia venimos realizando diversos cambios, adaptaciones y agregados para tener una base más moderna, potente y acorde a nuestras necesidades.

Recursos:

- Paquete de Nachos usado en la cátedra
- Sitio web oficial de Nachos
- Mapa de ruta de Nachos

1.2. ¿Cómo hacerse con Nachos?

La versión de Nachos que usamos en la cátedra se encuentra disponible en el repositorio Subversion del que disponemos en la carrera. En el directorio `Public` de la materia hay un subdirectorio `nachos` que contiene el paquete que hay que bajar. Dentro del mismo, tenemos subdirectorios por años, con las distintas versiones de Nachos que hemos ido usando para el dictado de cada año.

Dentro de `2023`, el subdirectorio `nachos-unr23a` contiene todo el árbol de archivos de Nachos. El archivo `.tar.xz` es un paquete que contiene este mismo árbol.

Como paso inicial para trabajar con Nachos, cada alumno o grupo debe hacerse una copia del directorio `nachos/2023/nachos-unr22a`. La sección que sigue brinda más información al respecto.

Como dato adicional, si se necesita el `.tar.xz`, este también está disponible en el Campus Virtual de la FCEIA.

1.3. ¿Cómo organizar los archivos en Subversion?

Antes que nada, se recuerda que está disponible la guía de Subversion provista en Arquitectura del Computador para referencia de comandos y de los aspectos generales del trabajo con Subversion.

Todas las resoluciones de las planchas deben subirse al repositorio Subversion de la carrera. El directorio correspondiente a Sistemas Operativos II es `R-412`. Dentro del mismo, hay un subdirectorio `Alumnos`, con subdirectorios para cada año.

La ruta completa es:
`https://svn.dcc.fceia.unr.edu.ar/svn-no-anon/lcc/R-412/Alumnos/2023/`

Cada alumno o grupo debe crearse un directorio dentro de `R-412/Alumnos/2023`. El nombre del directorio debe incluir los apellidos de cada integrante. Dentro de este, el contenido se organizará siguiendo una estructura de subdirectorios determinada: un directorio `trunk`, que contendrá el árbol de Nachos donde se hará el trabajo continuo a lo largo del cursado, y otro `tags`, donde se copiará el contenido de `trunk` para las entregas de cada práctica.

Con `svn checkout`, `svn mkdir` y `svn commit` se puede preparar esta jerarquía, por ejemplo:

```
$ svn mkdir \
    https://svn.dcc.fceia.unr.edu.ar/svn-no-anon/lcc/R-412/Alumnos/2023/foobar
$ svn checkout \
    https://svn.dcc.fceia.unr.edu.ar/svn-no-anon/lcc/R-412/Alumnos/2023/foobar
$ cd foobar
$ svn mkdir tags trunk
$ svn commit
```

La copia de Nachos se debe crear a partir del árbol disponible en el repositorio de Subversion, usando el comando `svn copy`. Por ejemplo:

```
$ svn copy \
    https://svn.dcc.fceia.unr.edu.ar/svn-no-anon/lcc/R-412/Public/nachos/2023/nachos-unr22a \
    https://svn.dcc.fceia.unr.edu.ar/svn-no-anon/lcc/R-412/Alumnos/2023/foobar/trunk
```

Una vez completado el código en `trunk` para una entrega y hechos los *commits* correspondientes, se lo copia a `tags` con un subdirectorio propio. Por ejemplo:

```
$ cd foobar
$ svn copy trunk tags/plancha1-entrega1
$ svn commit
```

1.3.1. Pautas y recomendaciones

1. Cada vez que realice una entrega, cree una etiqueta. Una etiqueta, en los sistemas de control de versiones, es un nombre asociado a una revisión en particular de un repositorio. Por referirse a una revisión dada, el contenido accedido mediante esa etiqueta nunca cambia, a diferencia del que se accede normalmente.

   Subversion no implementa el concepto de etiquetas, así que hay que simularlo a mano. Para esto, cree un directorio `tags`. Dentro del mismo, cada vez que desee definir una etiqueta, cree un subdirectorio con su nombre y dentro del mismo copie los archivos que conformen su contenido. Asegúrese de nunca modificar nada de este contenido.

   A la hora de corregir, los docentes de la materia miraremos solamente lo que esté en el directorio `tags`.

   Subcomandos de Subversion útiles: `svn mkdir`, `svn copy`.

2. Al mismo nivel de `tags`, puede crear también un directorio `trunk`. En este podrá mantener el árbol de trabajo, donde vaya haciendo todos los cambios.

   Hacer esto es opcional, el Subversion está disponible para que lo aproveche y en la materia, por defecto, asumiremos que lo utilizará, pero si prefiere optar en cambio por algún repositorio externo de su agrado, puede hacerlo. Lo obligatorio es que, una vez terminada cada plancha, realice la entrega correspondiente en Subversion por medio de una etiqueta.

   Subcomando de Subversion útil: `svn mkdir`.

3. No suba archivos generados automáticamente. Nachos al compilarse genera diversos archivos, en su mayoría binarios y algunos también de texto. Estos no deberían replicarse en el repositorio: se trata de archivos que se pueden regenerar, así que ocupan espacio innecesariamente y esto impacta no solo en el uso de disco sino, más críticamente, en la velocidad de descarga.

   Tenga en cuenta además que usted nunca debería modificar estos archivos, porque los cambios serían muy propensos a perderse debido a una sobreescritura automática. Por esto no le afecta el hecho de no subirlos.

   Concretamente, evite subir lo siguiente:

   - `code/bin/coff2flat`
   - `code/bin/coff2noff`
   - `code/bin/disassemble`
   - `code/bin/*.o`
   - `code/filesys/Makefile.depends`
   - `code/filesys/nachos`
   - `code/filesys/switch*.s` (`s` minúscula)
   - `code/filesys/*.o`
   - `code/network/Makefile.depends`
   - `code/network/nachos`
   - `code/network/switch*.s` (`s` minúscula)
   - `code/network/*.o`
   - `code/threads/Makefile.depends`
   - `code/threads/nachos`
   - `code/threads/switch*.s` (`s` minúscula)
   - `code/threads/*.o`
   - `code/userland/filetest`
   - `code/userland/halt`
   - `code/userland/matmult`
   - `code/userland/shell`
   - `code/userland/sort`
   - `code/userland/tiny_shell`
   - `code/userland/*.coff`
   - `code/userprog/Makefile.depends`
   - `code/userprog/nachos`
   - `code/userprog/switch*.s` (`s` minúscula)

- `code/userprog/*.o`
- `code/vmem/Makefile.depends`
- `code/vmem/nachos`
- `code/vmem/switch*.s` (`s` minúscula)
- `code/vmem/*.o`

¿Cómo evitar esto de forma cómoda? La idea es no tener que borrar en ningún momento, del árbol local donde uno trabaja, los archivos a mano. Hay dos opciones:

a) Cuidarse de nunca hacer `svn add` sobre estos archivos; o en caso de hacerlo, borrarlos con `svn rm --keeplocal` antes de efectuar algún "commit".

b) Marcarlos como ignorados por Subversion, preferentemente usando las propiedades `svn:ignore` o `svn:global-ignores`.

Subcomandos de Subversion útiles: `svn rm`, `svn rm --keeplocal`, `svn propset`.

1.4. Búsquedas en el árbol de archivos

Al familiarizarse con código que escribió otra persona, frecuentemente hace falta localizar el archivo donde está cierta definición. Esto se agiliza mucho usando herramientas adecuadas.

En cualquier sistema Unix, el comando `grep` es de ayuda para esta tarea. Se invoca con la sintaxis:

```
$ grep <patrón> <archivo>...
```

También se puede invocar de forma recursiva sobre el directorio actual:

```
$ grep -r <patrón>
```

En algunos sistemas existe `rgrep`, definido como un alias de `grep -r`.

Para más información, consulte la página de manual, `grep(1)`:

```
$ man 1 grep
```

El manual en formato GNU Info, en caso de estar instalado, es una referencia extensa:

```
$ info grep
```

Existen alternativas más recientes a `grep` que mejoran diversos aspectos, si bien no son estándares ni ubicuas. También hay herramientas de etiquetado de identificadores (como `ctags` y `etags`), editores de texto y entornos de desarrollo integrados que incorporan sus propias funcionalidades de búsqueda equivalentes. Animamos a los alumnos a que investiguen estas herramientas y encuentren aquellas que mejor se integren con su forma de trabajo.

2. Ejercicios

Véase el documento *Plancha 1*.