Práctica 1: Introducción a Nachos

2020 – Sistemas Operativos 2

Licenciatura en Ciencias de la Computación

Entrega: viernes 30 de marzo

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 descargar 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:

1. Un archivo `nachos-ultimo.tar.xz`. Este es el paquete con la última versión, basta con descargar simplemente este archivo y no tener en cuenta lo demás.
2. Subdirectorios por años, con las distintas versiones de Nachos que hemos ido usando para el dictado de cada año (`nachos-ultimo.tar.xz` es, concretamente, una copia de la última versión del año actual: `2019/nachos-unr19c.tar.xz`).

El archivo que uno descarga de Nachos es un paquete, es decir, se trata de un archivo que incluye varios archivos adentro; y además, está comprimido. Se lo puede descomprimir

y desempaquetar con diversas aplicaciones, tanto con de interfaces gráficas como de línea de comandos. En particular, una opción es emplear el comando `tar`, así:

```
$ tar xJf nachos-ultimo.tar.xz
```

Esto genera un directorio `nachos` con todo el contenido.

Para saber más

`tar` es un comando clásico de sistemas Unix. En principio, es una herramienta que solo empaqueta, no comprime; genera un archivo con formato TAR y extensión `.tar`. Lo que se hace comúnmente, entonces, es generar un paquete y luego pasarlo por algún compresor. Los hay en varios formatos, XZ es uno que se viene popularizando desde 2009 en sistemas Unix libres por su excelente nivel de compresión; sus archivos tienen extensión `.xz`.

Para usar lo que hay dentro del paquete comprimido, se hace el proceso inverso: primero se descomprime, después se desempaqueta. El comando `tar` en sí se encarga del desempaquetado, sin embargo se le puede pasar una opción para que invoque automáticamente al descompresor y así hacer todo junto. Para `xz`, esta opción es `J`.

Más información en las páginas de manual de los comandos `tar` y `xz`:

```
$ man tar
$ man xz
```

1.3. ¿Cómo organizar los archivos en 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.

Cada alumno o grupo debe crearse un directorio dentro de `R-412/Alumnos/2019`. El nombre del directorio debe incluir los apellidos de cada integrante. Dentro de este, el contenido se organizará siguiendo la estructura de subdirectorios indicada por la cátedra. En particular, será necesario subir una copia propia de Nachos o al menos las partes modificadas del mismo.

La ruta completa es:
`https://svn.dcc.fceia.unr.edu.ar/svn-no-anon/lcc/R-412/Alumnos/2020/`

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/swtch.s`
   - `code/filesys/*.o`
   - `code/network/Makefile.depends`
   - `code/network/nachos`
   - `code/network/swtch.s`
   - `code/network/*.o`
   - `code/threads/Makefile.depends`
   - `code/threads/nachos`
   - `code/threads/swtch.s`
   - `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/swtch.s`

- `code/userprog/*.o`
- `code/vmem/Makefile.depends`
- `code/vmem/nachos`
- `code/vmem/swtch.s`
- `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`.

2. Ejercicios

Responda a las siguientes preguntas en base al código de Nachos. Las respuestas deben ser incluidas en un archivo de texto.

1. ¿Cuánta memoria tiene la máquina simulada para Nachos?
2. ¿Cómo cambiaría ese valor?
3. ¿De qué tamaño es un disco?
4. ¿Cuántas instrucciones de MIPS simula Nachos?
5. Explique el código que procesa la instrucción `add`.
6. Nombre los archivos fuente en los que figuran las funciones y métodos llamados por el `main` de Nachos al ejecutarlo en el directorio `threads` (hasta dos niveles de profundidad).
7. ¿Por qué se prefiere emular una CPU en vez de utilizar directamente la CPU existente?
8. ¿Qué efecto hacen las macros `ASSERT` y `DEBUG` definidas en `lib/utility.hh`?
9. Comente el efecto de las distintas banderas de depuración.
10. ¿Dónde están definidas las constantes `USER_PROGRAM`, `FILESYS_NEEDED`, `FILESYS_STUB` y `NETWORK`?
11. ¿Cuál es la diferencia entre las clases `List` y `SynchList`?
12. ¿En qué archivos está definida la función `main`? ¿En qué archivo está definida la función `main` del ejecutable `nachos` del directorio `userprog`?
13. ¿Qué línea de comandos soporta Nachos? ¿Qué efecto hace la opción `-rs`?
14. Modifique el ejemplo del directorio `threads` para que se generen 5 hilos en lugar de 2.
15. Modifique el ejemplo para que estos cinco hilos utilicen un semáforo inicializado en 3. Esto debe ocurrir solo si se define la macro de compilación `SEMAPHORE_TEST`.
16. Agregue al ejemplo anterior una línea de depuración que diga cuándo cada hilo hace un *P()* y cuándo un *V()*. La salida debe verse por pantalla *solamente si se activa la bandera de depuración correspondiente.*