Auxiliar 1

20 de octubre de 2011

1. Introducción a Pintos: Instalación

Para las tareas del curso se utilizará un pequeño sistema operativo creado para un curso análogo[1] de Stanford: Pintos. Éste está probado en equipos con Ubuntu y Debian; además, es necesario tener GNU binutils, Perl ($\geq$5.8) y make ($\geq$3.8). También es útil tener GDB instalado.

Como Pintos es un sistema operativo, debe correr sobre un procesador, etc, para lo cual se utilizarán emuladores. Pintos puede utilizar (según se especifica en su página) Bochs, QEMU o VMware Player. Por ahora probaremos con QEMU, que no requiere mayor configuración. Si se desea utilizar Bochs (el emulador que Pintos asume por defecto) se debe configurar de forma especial[2]. Puede que deseen usar Bochs porque algunas opciones de Pintos están sólo para este emulador.

Una vez descargado el archivo con Pintos, se extrae la carpeta `src/` en la ubicación que se desee.

- Instalar el paquete `qemu` (o correspondiente)
- Crear una carpeta para los ejecutables de Pintos en la ubicación que se desee, y copiar los siguientes archivos a esta carpeta:
  - `src/utils/backtrace`
  - `src/utils/pintos`
  - `src/utils/pintos - gdb`
  - `src/utils/pintos - mkdisk`
  - `src/utils/pintos - set - cmdline`
  - `src/utils/Pintos.pm`
- Agregar esta carpeta a la variable PATH (por ejemplo, ejecutar `PATH = $PATH : /home/.../eldirectorio/`)
- Crear una carpeta o escoger una ubicación a la cual copiar el archivo `src/misc/gdb - macros`
- Editar el archivo `pintos - gdb` copiado, editando la variable `GDBMACROS` para que apunte al archivo `gdb - macros` copiado en el paso anterior.
- Ejecutar el comando `pintos - gdb` sin argumentos. No debería "quejarse" con respecto a las macros (si lo hace, se puso mal la dirección en 4)
- Editar el Makefile en `src/utils/`, quedando de la siguiente forma (sólo se debiera agregar la línea "$(CC)..."):

[1] Pueden encontrar (mucha) información del curso en `http://www.scs.stanford.edu/11wi-cs140/`. Posiblemente sea un link bastante útil.

[2] Mayores detalles en `http://www.scs.stanford.edu/11wi-cs140/pintos/pintos_12.html#SEC167`.

```
all: setitimer-helper squish-pty squish-unix

CC = gcc
CFLAGS = -Wall -W
LDFLAGS = -lm
setitimer-helper: setitimer-helper.o
        $(CC) setitimer-helper.o $(LDFLAGS) -o setitimer-helper
squish-pty: squish-pty.o
squish-unix: squish-unix.o
clean:          rm -f *.o setitimer-helper squish-pty squish-unix
```

- Ejecutar make en `src/utils/`. Copiar el archivo `squish-pty` a la carpeta de ejecutables ya creada. Para usar VMware Player, copiar también `squish-unix`. Copiar `setitimer-helper` si se tiene una versión de Perl particularmente antigua (menor a la 5.8).
- Fin (?)

2. Introducción a Pintos: Ejecución

Para empezar (y para la primera tarea) Pintos es sólo el kernel (es decir, todavía no existen programas de usuario). Por esto, si se quiere ejecutar algo, se debe compilar *junto con el sistema operativo.* Esto es lo que se hace en la carpeta `src/threads`. Al ejecutar `make` en esta carpeta, se crea el directorio `build`, que contiene el kernel compilado. Desde `build` se puede ejecutar Pintos, por ejemplo, mediante el comando `pintos --qemu -- run alarm-priority`.

- La opción `--qemu` se pasa al script y especifica el emulador a utilizar.
- El doble guión "solo" es un separador entre los argumentos para el script y los argumentos para Pintos.
- Después del doble guión, `run alarm-priority` indica al sistema operativo que ejecute el comando `alarm-priority`.

¿De dónde sale `alarm-priority` y qué es? Como ya se dijo, todavía no existen "programas de usuario" en el kernel, por lo que fue compilado junto con el kernel al hacer `make`. Los programas compilados, por tanto, se listan allí. Los programas que se instalan son tests para las tareas (!). Los tests que se compilan en este caso son los presentes en `src/tests/threads`. Pueden modificar los tests si lo necesitan.