Organizacion de Computadoras (66.20)

Herramientas

Herramientas

- Compilador: GCC
- Emulador: GXemul
- Documentacion: LaTeX

Convenciones

- Las líneas que comienzan con 'hostOS$' o 'guestOS$' indican el prompt de un shell, en el host OS y guest OS respectivamente.

  Ejemplos:

  ```
  hostOS$ ./gxemul -e 3max -d netbsd-pmax.img
  guestOS$ gcc -Wall -o foo foo.c
  ```

- Las líneas que comienzan con '$' indican el prompt de un shell, pero no hacen referencia a host OS ni a guest OS.

- 'hostOS#', 'guestOS#' y '#' indican implicitamente el prompt para el usuario root.

GCC (GNU Compiler Collection)

- Compilador C (entre otros)
- Gratuito y open source
- Soporta múltiples arquitecturas (inclusive MIPS)
- Genera código assembly

GCC (cont.)

- Supongamos que myprog.c es el código fuente en C a compilar:

```
$ gcc -Wall -o myexec myprog.c
```

Donde:

- `-Wall`: activa todos los mensajes de warning
- `-o`: archivo de salida (en este caso, myexec)

GCC (cont.)

- Para detener al compilador justo después de generar el código assembly:

```
$ gcc -Wall -O0 -S -mrnames myprog.c
```

Donde:

- `-S`: detiene al compilador luego de generar el assembly
- `-mrnames` (solo para MIPS): indica al compilador que genere la salida utilizando nombre de registro en lugar de número de registro
- `-O0`: No aplica optimizaciones

  - Esto genera el archivo myprog.s con el assembly que gcc genera para myprog.c

GCC (cont.)

- Documentación:
  - Manual page

    $ man gcc
  - Documento Info

    $ info gcc
  - Documento HTML
    - En debian se dispone del paquete gcc-<version>-doc, que instala la versión HTML del manual de gcc en /usr/share/doc/gcc-<version>-doc/gcc.html.

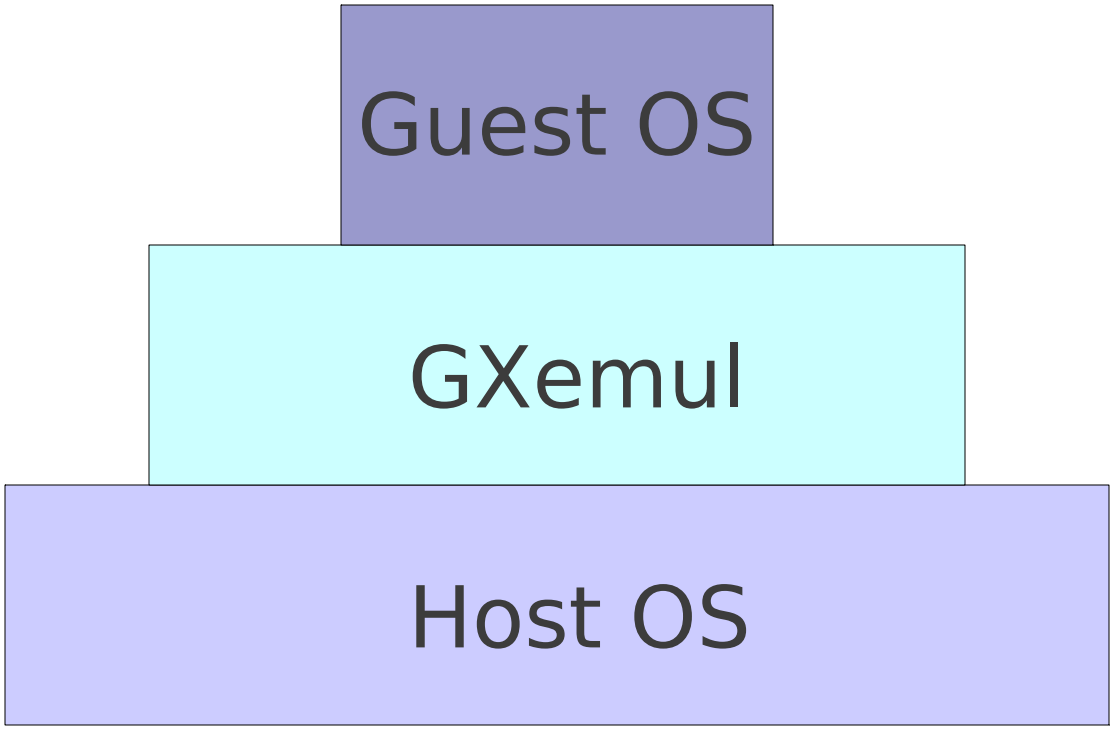
GXemul

- Gratuito y open source
- Emula múltiples arquitecturas (especialmente MIPS) a nivel instrucción, incluso con soporte de red (incipiente pero utilizable)
- Capaz de correr algunos OS sin modificaciones
  - Especialmente, NetBSD
  - Algunos otros Unix-like, pero menos probado

GXemul (cont.)

- Plataforma emulada
  - DECstation 5000/200 (pmax, a.k.a. “3max”) con procesador MIPS R3000.
- Host / Guest OS
  - Host OS: Sistema operativo “anfitrión” (es decir, sobre el cual corre gxemul)
  - Guest OS: Sistema operativo “invitado” (es decir, el que corre sobre gxemul)

GXemul (cont.)



GXemul (cont.)

Para el curso se proveen:

- Ejecutables “patrón” compilados estáticamente
  - gxemul: compilado sin soporte de X11
  - xgxemul: compilado con soporte de X11
- Imagen de disco “patrón”
  - netbsd-pmax.img: imagen de disco con instalación mínima de NetBSD 3.0 para plataforma pmax, con algunos programas “extra” de utilidad

GXemul (cont.)

- Ejecutables “patrón”
  - Para evitar utilizar las distintas versiones provistas por los posibles host OS
  - Compilados estáticamente para evitar problemas de interoperabilidad con los host OS
  - gxemul: sin soporte para X11
    - La consola que presenta gxemul es la misma desde la cual se corre en el host OS.
    - CTRL-C se implementa con CTRL-B

GXemul (cont.)

- Imagen “patrón”
  - Se provee una imagen patrón sobre la cual trabajaremos:
  - NetBSD/pmax 3.0
  - Configuración de red:
    - IP: 10.0.0.1 (cuidado con tener configurada esta IP en el host OS).
    - Mask: 255.255.255.0
    - Gateway: 10.0.0.254
    - Server DNS: 10.0.0.254

GXemul (cont.)

- Ejecución de gxemul sin soporte X11
  - Desde el directorio donde se instaló el gxemul y la imagen patrón, correr:

    ```
    hostOS$ ./gxemul -e 3max -d netbsd-pmax.img
    ```

    Donde:

    `-e 3max`: arquitectura 3max (a.k.a. pmax)

    `-d netbsd-pmax.img`: imagen de disco a utilizar (donde está instalado nuestro guest OS: NetBSD 3.0 para pmax)
  - Así se bootea la imagen del disco patrón

GXemul (cont.)

- Al final del proceso tendremos login prompt
- Ingresamos con:
  - usuario: root
  - password: orga6620
- Este gxemul presenta la misma consola que se utiliza para correrlo.
  - CTRL-C se reemplaza por CTRL-B (porque la primera se utiliza por GXemul para entrar en modo debug)
  - Solo se dispone de una consola muy limitada

GXemul (cont.)

- Ejecución de gxemul con soporte X11
  - Se puede usar el archivo “patrón” xgxemul
  - Adicionalmente, es probable que el host OS provea un gxemul compilado con soporte X11 (caso de debian sarge o etch, por ejemplo)
  - Con la opción “-x” le indicamos al gxemul que inicie una consola xterm en el host OS que usará como consola del guest OS

```
hostOS$ xgxemul -e 3max -d netbsd-pmax.img -x
```

GXemul (cont.)

- Tunel SSH hacia el guest OS
  - Es la mejor opción puesto que podemos abrir la cantidad de consolas que deseemos contra el guest OS, por ejemplo por SSH
  - Problema: el networking de GXemul es incipiente, y lo implementa con sockets hacia el host OS (es decir, no presenta una interfaz de red “real” en el host OS, sino que abre sockets TCP/IP).
  - Como consecuencia, no se pueden realizar conexiones entrantes de forma directa al guest OS

GXemul (cont.)

- Solución: Tuneles
  - Se puede implementar con ssh/sshd, lo cual requiere sshd instalado en el host OS (y ssh en el guest, lo cual está contemplado en la imagen patrón)
- Creación de un tunel SSH
  - Primero, necesitamos un usuario no privilegiado en host OS. Si no existe, root puede crearlo con:

```
hostOS# useradd -m gxemul
hostOS# passwd gxemul
... (prompt para ingresar password para gxemul)
```

  - Si el usuario existe, no es necesario crear a gxemul, pero nos referiremos a este como gxemul.

GXemul (cont.)

- Creamos en el host OS con el usuario root, un alias para la interfaz loopback (lo:0) con la IP 172.20.0.1:

```
hostOS# ifconfig lo:0 172.20.0.1
(el efecto no persiste ante reboots del host)
```

- Nos conectamos contra esa interfaz:

```
guestOS$ ssh -R 2222:127.0.0.1:22 gxemul@172.20.0.1
... (login de gxemul en host OS) ...
hostOS$
```

- Aquí tendremos un shell remoto contra host OS sobre la consola de gxemul (***que dejaremos de usar***)

GXemul (cont.)

- El sshd en el host OS crea un socket sobre la loopback y el puerto 2222, y las conexiones contra el mismo se forwardean por el tunel SSH hacia guest OS

– Desde host OS podemos iniciar shells remotos contra guest OS, abriendo terminales y ejecutando:

```
hostOS$ ssh -p 2222 root@127.0.0.1
... (login contra guest OS) ...
guestOS$
```

GXemul (cont.)

- Desde host OS podemos iniciar shells remotos contra guest OS, abriendo terminales y ejecutando:

```
hostOS$ ssh -p 2222 root@127.0.0.1
... (login contra guest OS) ...
guestOS$
```

GXemul (cont.)

- Transferencia de Archivos: SCP
  - Se puede utilizar la utilidad “scp” para la transferencia de archivos entre el host OS y el guest OS.
  - Utiliza el tunel SSH previamente creado
- Uso:

  $ scp -P2222 [-r] origen destino

  -r es opcional e indica que el origen debe copiarse de forma recursiva (directorios)

  origen o destino pueden ser locales o remotos (pero si uno es remoto el otro es local, y viceversa, no los dos remotos o locales al mismo tiempo)

GXemul (cont.)

- Ejemplo de uso de SCP con destino remoto y copia recursiva:

```
hostOS$ scp -P2222 -r /home/mi_usuario/66.20/tp0 root@127.0.0.1:/home/root
```

  Aquí se copia el directorio en /home/mi_usuario/66.20/tp0 hacia NetBSD (por el tunel) en el directorio /home/root (usuario root del netbsd)

- Ejemplo: SCP con origen remoto:

```
hostOS$ scp -P2222 root@127.0.0.1:/home/root/tp0/tp0.c /home/mi_usuario/66.20/tp0
```

  Aquí se copia un único archivo de NetBSD hacia Linux

LaTeX

- Permite concentrarse en el contenido del documento en lugar de la forma del mismo
- Formato abierto y de texto (se pueden mantener los documentos con CVS)
- Resultados muy profesionales
- Templates tipo “paper”

LaTeX (cont.)

Ejemplo de documento LaTeX básico:

```
\documentclass[a4paper,10pt]{article}
\usepackage[spanish]{babel}
\usepackage[latin1]{inputenc}
\begin{document}
  \section{Una sección}
    Texto de ``Una sección''
    \subsection{Una subsección}
      Texto de ``Una subsección''
  \section{Otra sección}
    Texto de ``Otra sección''
\end{document}
```

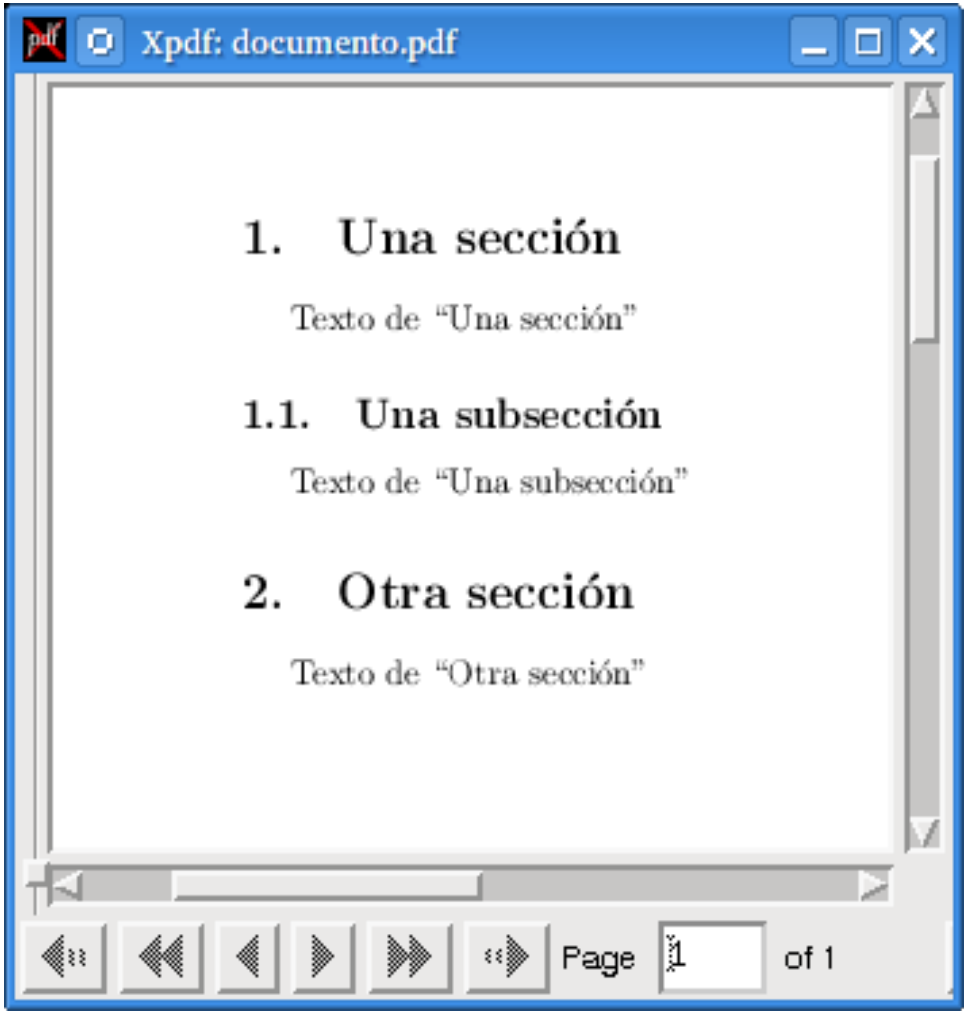
LaTeX (cont.)

- Procesamiento del documento TeX
  - Al procesar un documento TeX, LaTeX genera varios archivos, entre ellos un archivo DVI (DeVice Independent)
  - El archivo DVI puede convertirse a varios formatos “finales”, como PostScript (ps) o PDF

```
hostOS$ latex document.tex
... mensajes de procesamiento LaTeX ...
hostOS$ dvipdf document.dvi
```

    - Esto último genera el archivo document.pdf

LaTeX (cont.)


Xpdf: documento.pdf
1. Una sección
Texto de "Una sección"
1.1. Una subsección
Texto de "Una subsección"
2. Otra sección
Texto de "Otra sección"
Page 1 of 1

LaTeX (cont.)

- Inclusión de archivos EPS (Encapsulated PostScript)
  - Se pueden incluir gráficos en archivos EPS dentro de un documento LaTeX, utilizando por ejemplo el paquete graphicx
  - Se proporcionará un documento patrón para la realización de los TP
- Documentación
  - “The Not So Short Introduction To LaTeX”

Sitios de la materia

- http://materias.fi.uba.ar/6620/
- http://groups.yahoo.com/group/orga-comp/

- URL Imagen patrón / gxemul:

  https://docs.google.com/file/d/0B93s6e6NY_j1ajd2Y0JBR0hudlU/edit?usp=sharing