INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

COMPUTER SYSTEMS LABORATORY

Fall 2011

Stage -1

External software

- VMware Player *http://www.vmware.com/download/player/*
- Ubuntu-10.04-x86 *http://www.deetc.isel.ipl.pt/programacao/psc/Ferramentas/Ubuntu-10.04-x86-beta.zip*
- Bochs 2.4.6 *http://sourceforge.net/projects/bochs/files/bochs/2.4.6/bochs-2.4.6.tar.gz/download*
- FreeDOS 1.0 *http://www.finnix.org/Balder*

Setting up the basic environment

The following notes indicate how to setup the recommended environment. You may use a different Linux system if you want to, but the more you deviate from the recommended environment, the less we'll be able to help you with configuration issues. In any case, recent Ubuntu variants should cause less trouble than other options.

If you don't have the *Ubuntu-10.04-x86* virtual machine already installed:

- Install VMware Player
- Unzip *Ubuntu-10.04-x86-beta.zip* into some convenient location (e.g.: `C:\LinuxVM\`)
- Run *Ubuntu-10.04-x86.vmx* and select "*I copied it*" when the question pops up
- VMware Player may complain about VMware Tools not being up to date, but you can ignore that for now
- Finally, check that you are able to:
  - resize the VMware Player window
  - move the mouse in and out of the VMware Player window
  - drag-and-drop files between the Ubuntu-10.04-x86 VM and your host system

Installing Bochs

Bochs is an open source IA-32 PC emulator with an integrated low-level debugger. Bochs debugging capabilities will be particularly useful during the first stages of this course, where the setup is too hostile for a common debugger.

Ubuntu already has a pre-packaged version of Bochs, but the internal debugger was left out of it, so we need to follow the hard path. In the unlikely event of something going wrong with the instructions we're providing you, another option is to proceed with Ubuntu's default Bochs package. You'll probably miss the debugger, though.

- Place `bochs-2.4.6.tar.gz` at `~/lsc/`, open a terminal window, and execute:
  - `cd ~/lsc`
  - `tar xzvf bochs-2.4.6.tar.gz`
  - `cd bochs-2.4.6`
  - `geany lsc-conf &`

- In the open text editor, place the following text, then save the file and leave the editor:

```
#!/bin/sh
./configure --enable-smp \
            --enable-cpu-level=6 \
            --enable-acpi \
            --enable-all-optimizations \
            --enable-x86-64 \
            --enable-pci \
            --enable-vmx \
            --enable-debugger \
            --enable-disasm \
            --enable-debugger-gui \
            --enable-logging \
            --enable-vbe \
            --enable-fpu \
            --enable-sb16 \
            --enable-cdrom \
            --enable-x86-debugger \
            --enable-iodebug \
            --disable-plugins \
            --disable-docbook \
            --with-x11
```

- Go back to the terminal window, make sure you're at `~/lsc/bochs-2.4.6` and execute:
  - `chmod 755 lsc-conf`
  - `sudo apt-get install xserver-xorg-dev xorg-dev libgtk2.0-dev`
  - `./lsc-conf`
  - `make`
  - `sudo make install`
  - `cp .bochsrc ..`
  - `cd ..`
  - `geany .bochsrc &`
- Make sure that `.bochsrc` has the following options set as shown below:
  - `display_library: x, options="gui_debug"`
  - `memory: guest=32, host=4`
  - `floppya: 1_44=/home/isel/lsc/disks/lsc.fd, status=inserted`
  - `#ata0: enabled=0, […]`
  - `#ata1: enabled=0, […]`
  - `#ata2: enabled=0, […]`
  - `#ata3: enabled=0, […]`
  - `#ata0-master: […]`
  - `#ata0-slave: […]`
  - `boot: floppy`
  - `#boot: disk`
  - `debugger_log: debugger.out`
- Save the file (`.bochsrc`) and close the editor
- Place the virtual floppy from the previous lab (Overture) in `~/lsc/disks/` naming it `lsc.fd` (alternatively, use the attached file `1986.fd`)
- Back at the terminal, make sure the current directory is `~/lsc` and execute `./bochs`
- Set up a breakpoint by typing "`b 0x7c00`" (without the quotes) in the command box of Bochs' internal debugger – that's the text box at the bottom of the debug window
- Continue execution with command `c` and execute your code step-by-step

Bochs keyboard layout

Bochs is a PC emulator which runs on top of some host operating system. In our case, we're using a Ubuntu Linux system, where Bochs runs atop the X Window System, which generates key events already translated by some key map.

Bochs translates X Windows key events back into low level key codes assuming, by default, that X Windows is using a north-american key map, which is not our case. To get correct low level key codes (called scan codes) in Bochs, copy the attached key map (*x11-pc-pt.map*) to */usr/local/share/bochs/keymaps/* and configure the following line in *.bochsrc* :

```
keyboard_mapping: enabled=1, map=/usr/local/share/bochs/keymaps/x11-pc-pt.map
```

Creating the virtual hard disk image

PC emulators and virtualizers support a variety of file formats for virtual hard disk images. The raw/flat file format is the most widely supported, making it relatively easy to use the same virtual hard disk image file across systems.

Creating a raw/flat virtual hard disk image for Bochs

Bochs accepts raw/flat image files as virtual hard disks, as long as the disk geometry is specified. An image file may be easily generated with dd.

1. Calculate an appropriate disk geometry (cylinders, heads, sectors) for the desired disk size. For a small disk (up to about 500MiB), you may fix heads at 16, sectors at 63, and calculate cylinders by rounding up the result of **`cylinders`** `= sizeMB * 2.0317`, where `sizeMB` is the total disk size in megabytes. For bigger disks, read this first.
   Example: for a disk size of 63MiB, **`cylinders`** `= 63 * 2.0317 = 127.9971`, and therefore disk geometry will be **`(C,H,S)`** `= (128,16,63)`
2. Obtain total number of sectors for the calculated geometry, with the simple formula **`total_sectors`** `= cylinders * heads * sectors`. Then get the exact disk size (in bytes) with **`size`** `= 512 * total_sectors`.
   Example: for a disk geometry of `(C,H,S) = (128,16,63)`, we get **`total_sectors`** `= 128 * 16 * 63 =` **`129024`**, and total disk size is **`size`** `= 512 * 129024 =` **`66060288`**, which is 63.0MiB.
3. In your Linux system, use `dd` to create a zeroed raw/flat disk image file with `total_sectors` blocks of 512 bytes.
   Example: `dd if=/dev/zero of=`**`lsc-hd63-flat.img`** `bs=512 count=`**`129024`**
4. Don't forget to place the file in the central directory for virtual disk images.
   `mv lsc-hd63-flat.img ~/lsc/disks`

The new virtual hard disk file **lsc-hd63-flat.img** can now be used in **Bochs** with the following lines in `.bochsrc`:

```
ata0: enabled=1, ioaddr1=0x1f0, ioaddr2=0x3f0, irq=14

# in a single line:
ata0-master: type=disk, path="/home/isel/lsc/disks/lsc-hd63-flat.img",
             mode=flat, cylinders=128, heads=16, spt=63,
             translation=none, biosdetect=auto, model="LSC HD"
```

Partitioning the virtual hard disk with FreeDOS fdisk tool

In order to have a partition table fully compatible with FreeDOS, we'll use its `fdisk` tool to partition the virtual hard disk.

1. The attached `balder10.fd` file is a floppy disk image containing Balder 10, a particular distribution of FreeDOS 1.0. Place this file in `~/lsc/disks/`, naming it `lsc.fd`.
2. Make sure that you have properly configured Bochs to use the virtual hard disk created in the previous section, while still booting from floppy.
3. Start Bochs and wait for the `A:\>` prompt. You may accept the default option in the intermediate menu (`3 – Start FreeDOS for 386 + FDXMS`).

4. At the `A:\>` prompt, run `fdisk`
    a. Do NOT enable support for FAT32
    b. Create a primary DOS partition with 21MiB (NOT with maximum available size)
    c. Create another primary DOS partition with 42MiB (all the remaining space)
    d. Set partition #1 as active
    e. Leave `fdisk` and reset Bochs

Installing FreeDOS in the first partition

For the purposes of this course, a trivial installation of FreeDOS is enough, so we'll just place the contents of the Balder 10 floppy disk in the first partition of the virtual hard disk.

1. With Bochs booted from the Balder 10 floppy:
    a. `A:\> format c: /S`
    b. `A:\> xcopy *.* c:\ /D /E /H /S`
    c. `A:\> edit c:\autoexec.bat`
        i. Go to the menu *Search* > *Replace…*
        ii. Replace ALL occurrences of **`A:`** with **`C:`**
        iii. Save and exit
2. Close Bochs
3. Edit `.bochsrc` to boot from disk instead of floppy
    ```
    #boot: floppy
    boot: disk
    ```
4. Run Bochs again. After the boot sequence, you should get the familiar `C:\>` prompt.

Checkpoint:

At this point, the virtual hard disk image has two partitions. The first partition has FreeDOS 1.0 installed, while the second is unformatted and empty. The partition table is stored at the master boot record, which also has bootstrap code placed there during the installation of FreeDOS.

The FreeDOS partition won't really be of much use during the course, except that it should remain intact and bootable, no matter what we do on the rest of the disk.

Using the virtual hard disk under VMware and/or Virtual Box

In order to use a raw/flat virtual hard disk image under VMware, create a text file based in the following content, and give it a *.vmdk* extension. This *.vmdk* can be used both in VMware and Virtual Box virtual machines.

```
# Disk DescriptorFile
version=1
CID=fffffffe
parentCID=ffffffff
createType="monolithicFlat"

# Extent description
RW 129024 FLAT "lsc-hd63-flat.img" 0

# The Disk Data Base
#DDB

ddb.virtualHWVersion = "3"
ddb.geometry.cylinders = "128"
ddb.geometry.heads = "16"
ddb.geometry.sectors = "63"
ddb.adapterType = "ide"
```