Written by: Jonathan Tjioe
Last Updated: 2010-10-25

Introduction
I spent several days trying to get DiskSim 3.0 running and had several issues. Through much effort, help from users on the DiskSim email list, Mahmood's guide, and Ajay's guide, I was able to get it running. When I first started trying to use DiskSim, I wished that there was a detailed, step-by-step guide that told me which OS to use, which packages to add, what to modify, etc. What follows is my attempt at creating that guide.

Helpful Links
DiskSim Users Email list
https://sos.ece.cmu.edu/mailman/listinfo/disksim-users

Ajay's Guide for DiskSim
http://www.cs.rice.edu/~gulati/

Original DiskSim Website
http://www.pdl.cmu.edu/DiskSim/

Environment
With all the different Linux distributions out there, every developer's environment will be a little bit different. But for the sake of this guide, here is what I know works and what I have setup. I will be installing Ubuntu 10.10 (32-bit) in a virtual machine using Parallels Desktop 6.0 on my MacBook Pro 2.66GHz i7, which is x86 based and running OS X 10.6.4. Of course, you should be able to virtualize Ubuntu on any x86 based Virtualization software and it should work fine. However, to my knowledge, DiskSim 3.0 does not work on 64-bit hardware. This may not hold true for the latest version, but I'm pretty sure 3.0 only works on 32-bit.

Software
Parallels Desktop 6.0 (Mac) or some other virtualization software
Ubuntu 10.10 (32-bit)
gcc-3.3 (3.3.6)
g++-3.3 (3.3.6)
bison 2.4.1
flex 2.5.4
DiskSim3.0 (packaged by Ajay)

Procedures
High-level Instructions
1. Install Parallels Desktop 6.0 (Mac) or some other virtualization software
2. Install Ubuntu in a Virtual Machine (VM)
3. Downgrade gcc and g++ and add other required software
4. Extract Ajay's tar file and modify accordingly

Written by: Jonathan Tjioe
Last Updated: 2010-10-25

5. Test out DiskSim 3.0

Detailed Instructions
Note: I won't cover installing a VM as you can use any virtualization product similar to Parallels Desktop, VMWare workstation, etc. However, I would make sure you setup the network card of your VM to be NAT. This will allow Internet access from the VM, which will be important to get the right packages using Ubuntu's Synaptic Package Manager. Also, it will probably make your life easier if you install Parallels Tools, VMWare Tools, or whatever your product's virtualization tools are called.

1. Download Ubuntu 10.10 from here:
   http://www.ubuntu.com/
2. Install Ubuntu in a VM.
3. Verify current version of gcc is not 3.3.6:
   `gcc -v`
4. `sudo gedit /etc/apt/sources.list` and enter the following lines at the end of the file:

```
## All officially supported packages, including security- and other updates
deb http://archive.ubuntu.com/ubuntu dapper main restricted
deb http://security.ubuntu.com/ubuntu dapper-security main restricted
deb http://archive.ubuntu.com/ubuntu dapper-updates main restricted

## The source packages (only needed to recompile existing packages)
deb-src http://archive.ubuntu.com/ubuntu dapper main restricted
deb-src http://security.ubuntu.com/ubuntu dapper-security main restricted
deb-src http://archive.ubuntu.com/ubuntu dapper-updates main restricted

## All community supported packages, including security- and other updates
deb http://archive.ubuntu.com/ubuntu dapper universe multiverse
deb http://security.ubuntu.com/ubuntu dapper-security universe multiverse
deb http://archive.ubuntu.com/ubuntu dapper-updates universe multiverse

## The source packages (only needed to recompile existing packages)
deb-src http://archive.ubuntu.com/ubuntu dapper universe restricted
deb-src http://security.ubuntu.com/ubuntu dapper-security universe restricted
deb-src http://archive.ubuntu.com/ubuntu dapper-updates universe restricted
```

5. Verify that Update Manager still functions properly:
   System > Administration > Update Manager

   Note: Hidden characters or improperly formatted/encoded text can cause the Update Manager to return an error. If an error comes up, edit the /etc/apt/sources.list file again and enter the text manually.

6. Once Update Manager can be opened successfully, close it. Open up Synaptic Package Manager:
   System > Administration > Synaptic Package Manager

7. Click Reload to ensure all repositories show the latest software packages. At the "Could not download all repository indexes" prompt, click Close.
8. Search for and mark the following packages for installation:
   gcc-3.3 base
   gcc-3.3-doc
   gcc-3.3 (3.3.6)
   g++-3.3 (3.3.6)
   bison (2.4.1)
   flex-old (2.5.4a-8)
   flex-old-doc (2.5.4a-8)

   Note: If the packages do not show up, try Reloading (Step 7) again.

   Note: Synaptic Package Manager automatically calculates dependencies between software packages. So if you are prompted to install other packages due to dependencies, install those as well.

   After selecting the desired software packages, click Apply 2 times. On the "Changes Applied" box, click Close. Then Close Synaptic Package Manager.

9. Shift the priority of the older gcc-3.3.6 to be above gcc-4.4.5 which came preinstalled with Ubuntu 10.10:
   ```
   sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-3.3 60 --
   slave /usr/bin/g++ g++ /usr/bin/g++-3.3

   sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-4.4 40 --
   slave /usr/bin/g++ g++ /usr/bin/g++-4.4
   ```

10. Verify that the default gcc is now 3.3.6:
    ```
    update-alternatives --config gcc

    gcc -v
    ```

11. Verify that the version of flex (2.5.4) and bison (2.4.1) are correct:
    ```
    bison --version
    flex --version
    ```

12. From the Ubuntu VM, download Ajay's tar file:
    http://www.cs.rice.edu/%7Egulati/tools/disksim-3.0-ajay.tar.gz

13. Extract the file to your home directory:
    ```
    cp <Download Path>/disksim-3.0-ajay.tar.gz ~
    cd ~
    ```

```
gunzip disksim-3.0-ajay.tar.gz
tar –xvf disksim-3.0-ajay.tar
```

14. Modify the newly extracted directory of DiskSim 3.0 according to the instructions below:

    Note: Ajay's original instructions can be found here: ~/disksim-3.0/HOW-TO-COMPILE.Ajay. I've made a few updates and the modified instructions are what follow.

    Note: For all entires that have "/home/mooncold", replace this with your path to the disksim-3.0 directory.

    <u>Compiling libddbg</u>
    ```
    cd /home/mooncold/disksim-3.0/libddbg
    make all
    ```

    <u>Compiling libparam</u>
    ```
    cd /home/mooncold/disksim-3.0/libparam

    #update the LIBDDBG_PREFIX in the .paths.in file to be the path
    LIBDDBG_PREFIX=/home/mooncold/disksim-3.0/libddbg

    cp .paths.in .paths
    make
    ```

    <u>Compiling diskmodel</u>
    ```
    cd /home/mooncold/disksim-3.0/diskmodel

    #update the LIBPARAM_PREFIX and LIBDDBG_PREFIX in the .paths.in file
    export LIBPARAM_PREFIX=/home/mooncold/disksim-3.0/libparam
    export LIBDDBG_PREFIX=/home/mooncold/disksim-3.0/libddbg

    cp .paths.in .paths
    make
    ```

    <u>Compiling the src directory</u>
    ```
    cd /home/mooncold/disksim-3.0/src

    #update the LIBPARAM_PREFIX and LIBDDBG_PREFIX in the .paths.in file
    export LIBPARAM_PREFIX=/home/mooncold/disksim-3.0/libparam
    export LIBDDBG_PREFIX=/home/mooncold/disksim-3.0/libddbg
    export DISKMODEL_PREFIX=/home/mooncold/disksim-3.0/diskmodel

    cp .paths.in .paths
    make
    ```

15. Verify that disksim is running properly.
    ```
    cd /home/mooncold/disksim-3.0/valid
    ./runvalid
    ```

Note: The rms values for the hardware and simulation results should be very similar. If it is, this means that DiskSim compiled successfully.

16. Run a sample simulation. From the valid directory, run the following:

```
../src/disksim barracuda.parv test.out ascii 0 1
```

Note: The output will be in test.out.