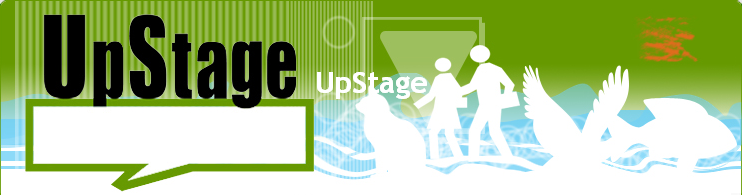
UpStage Developer Handbook

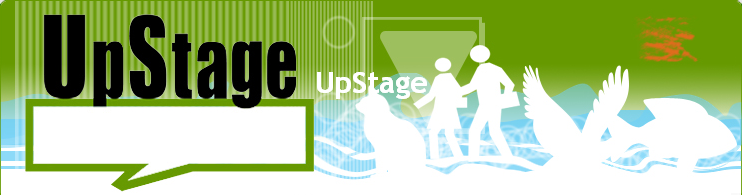
**2014**

**AUT UpStage Team 2014**

**1/1/2014**



**Developer Handbook History**

******

# History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Changes made | Date | Who |
| 0.1 | Initial Draft |  | Heath Behrens |
| 0.2 | Added sections for installing voices, added creating daemon server and upstage commands. | 06/07/2011 | Heath Behrens |
| 0.3 | Added Sections for Linux commands | 08/07/2011 | Heath Behrens |
| 0.3.1 | Cleaned up a bit | 09/07/2011 | Heath Behrens |
| 0.3.2 | Added more Linux commands namely basic command usage | 09/07/2011 | Heath Behrens |
| 0.4 | Added coding standards. | 10/07/2011 | Heath Behrens |
| 0.5 | Added compiling Actionscript 3 and 2 clients under Linux | 11/07/2011 | Heath Behrens |
| 0.6 | Added Cover page, TortoiseSVN Tutorial and setup ftp server | 14/07/2011 | Heath Behrens |
| 0.7 | Added terminal update command for updating a server from svn. | 07/09/2011 | Corey Robb  Heath Behrens |
| 0.8 | Added section “How voices work”, Added section for debugging. | 19/09/2011 | Corey Robb  Heath Behrens  Karena Goh |
| 0.9 | Added more details to the coding standards to include PyLint usage. | 05/10/2011 | Heath Behrens |
| 1.0 | Added content about Ant, updated installation section, added stuff about using Git, and removed content in coding standards section and referred to a separate document | 31/10/2013 | Nitkalya Wiriyanuparb |

Table of Contents

[History 1](#_Toc370990313)

[Version Control (Subversion / Git) 5](#_Toc370990314)

[Git Commit Protocol 10](#_Toc370990315)

[Versioning 10](#_Toc370990316)

[Ant Build Automation 11](#_Toc370990317)

[Installing UpStage 12](#_Toc370990318)

[Dependencies 12](#_Toc370990319)

[Quick start a server daemon 14](#_Toc370990320)

[Installation using Debian package 15](#_Toc370990321)

[Compiling ActionScript 3 Client 16](#_Toc370990322)

[Voices 18](#_Toc370990323)

[How Voices work 18](#_Toc370990324)

[Installing Voices 19](#_Toc370990325)

[Installing Enhanced MBROLA Voices 20](#_Toc370990326)

[CMU Arctic Voices 21](#_Toc370990327)

[Testing Festival 23](#_Toc370990328)

[Managing UpStage 24](#_Toc370990329)

[UpStage Directory Structure 25](#_Toc370990330)

[Introduction to Linux 26](#_Toc370990331)

[Linux File System Basics 26](#_Toc370990332)

[Installing Applications 27](#_Toc370990333)

[Simple Bash Commands 28](#_Toc370990334)

[More complex Commands 34](#_Toc370990335)

[Basic Command Usage 34](#_Toc370990336)

[SSH (Secure Shell) 36](#_Toc370990337)

[Development Environment 37](#_Toc370990338)

[Debugging 37](#_Toc370990339)

[Coding Standards 37](#_Toc370990340)

[PyLint 38](#_Toc370990341)

[Pep8 39](#_Toc370990342)

[Language Basics 40](#_Toc370990343)

[Python Basics 40](#_Toc370990344)

[ActionScript Basics 40](#_Toc370990345)

[AUT UpStage Server (WT501) 41](#_Toc370990346)

[Setting up vSFTP 41](#_Toc370990347)

[Enabling uploading 41](#_Toc370990348)

[Local user logging 42](#_Toc370990349)

[Anonymous logging 42](#_Toc370990350)

[Chroot jail 42](#_Toc370990351)

[Flash Tracer (Debugging Flash) 44](#_Toc370990352)

[Linux Install: 44](#_Toc370990353)

The document outlines the information required by a new UpStage developer and is intended for use by AUT UpStage students as a simple guide book to UpStage development. As such for the most part the information contained herein will pertain to the development version of UpStage which at the time of writing is 3.0.

This document should be updated on a regular basis and any disparities modified so that the manual remains up to date and accurate at all times.

**UpStage GitHub Repository:** <https://github.com/AUTUpStageTeam/UpStage2013>

**UpStage GitHub Issues:** <https://github.com/AUTUpStageTeam/UpStage2013/issues?state=open>

**UpStage Homepage:** <http://upstage.org.nz/blog/>

**UpStage Developer Blog:** <http://upstage.org.nz/blog/?page_id=1553>

**UpStage AUT Email:** [aut.upstage.team@gmail.com](mailto:aut.upstage.team@gmail.com)

**UpStage AUT Local SVN:** svn://elena.aut.ac.nz/home/upstage/svn

**UpStage User Manual:** <http://flossmanuals.net/> Under ‘Performance’ section

**UpStage Server Manual:** <https://github.com/AUTUpStageTeam/upstage2013/wiki/Server-Manual>

Old Links

**UpStage TRAC:** <http://sourceforge.net/apps/trac/upstage/>

**Sourceforge URL:** <http://sourceforge.net/projects/upstage/>

# Version Control (Subversion / Git)

UpStage uses Subversion and Git for version control, for now we will focus on Git. Let’s start with a simple how to, these are the basic commands and applications one can use to interact with Git.

The UpStage team uses a subversion repository hosted on Sourceforge.net for version control which is located at:

<https://github.com/AUTUpStageTeam/UpStage2013>

No login information is required to checkout code, however a GitHub user account is required in order to commit changes to the repository. See the Password document for AUT GitHub account credentials.

Linux

#### Git

Below outlines how to use the terminal to perform git functions in Linux.

To check out from the Git repository use the following command:

*git clone https://github.com/AUTUpStageTeam/UpStage2013*

The above command would checkout the latest revision of the master branch to the current directory and create a folder named UpStage2013.

It is recommended that you fork the main repository and work on your own repository, but you have to learn about ‘git remote’, ‘git fetch’, and ‘git merge’ to work with your forked repository.

Useful Git Tutorial/Book Links:

* <http://try.github.io/levels/1/challenges/1>
* <http://learn.github.com/>
* <http://git-scm.com/book>

To commit any changes simply change directory to upstage and run:

*git commit –m “A useful message”*

If you want to track a new file or add a file to the staging area use the following command:

*git add “filename”*

For more information on the commands that are available the Git Help file can be accessed by typing:

*git help*

To update a current working version:

*git pull*

*(assuming you are currently in the version directory, Internet is connected and port forwarding is on)*

#### Subversion

There is an application called <http://www.rabbitvcs.org/> which provides a simple user interface to Subversion. The application plugs into nautilus allowing similar functions to that of <http://tortoisesvn.net/> for Windows.

[RabbitVCS](http://www.rabbitvcs.org/) can easily be installed using the chosen Linux distributions package manager.

Use the following steps in Ubuntu:

*sudo add-apt-repository ppa:rabbitvcs/ppa*

*sudo apt-get update*

*sudo apt-get install rabbitvcs-core rabbitvcs-nautilus rabbitvcs-gedit rabbitvcs-cli*

Because of its similarity to [TortoiseSVN](http://tortoisesvn.net/) one can follow the windows guide to Subversion to use [RabbitVCS](http://www.rabbitvcs.org/).

Windows

#### Git

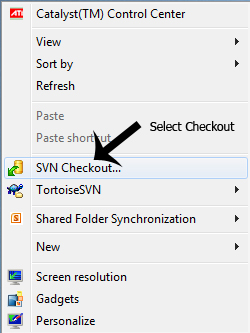
GitHub provides a client application for Git, named GitHub for Windows. It can be downloaded here: <http://windows.github.com/>. The application comes with a shell program, so commands listed above (under the Linux section) can also be used there.

#### Subversion

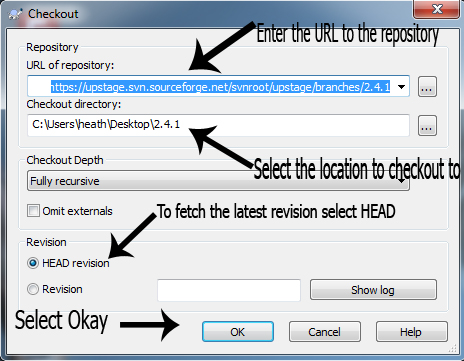
Using subversion in a windows environment is simple enough using the <http://tortoisesvn.net/> once downloaded and installed repositories can be checked out and managed through windows explorer.

Below is a simple walkthrough of checking out code and committing modified code to the repository using [TortoiseSVN](http://tortoisesvn.net/).

Figure 1 Context Menu

**Step 1:**

Code can be checked out from anywhere within windows explorer. For example right clicking on the desktop and selecting “checkout” from the context menu will start the process.



**Step 2:**

Figure 2 Checkout View

Now we need to enter the URL to the repository we want to check out, in our case that is the [UpStage Sourceforge URL](https://upstage.svn.sourceforge.net/svnroot/upstage/branches/2.4.1). The checkout directory is the location where the checked out code will be stored; in this case we selected the desktop so the code will be stored in a folder on the desktop named *2.4.1*.

Unless you want to check out a particular revision of the code base selecting HEAD (Which is default) will check out the latest revision.

**Step 3:**

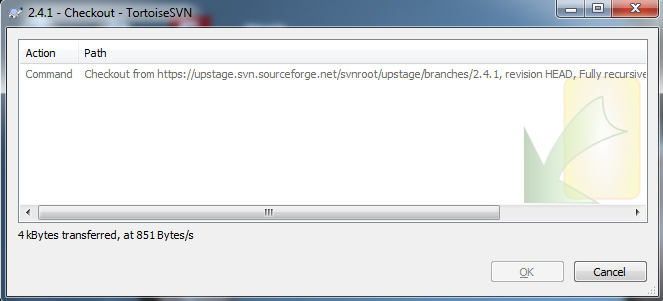
**** Depending on the amount of code and the size of the repository, it may take some time to process. Once the code has been successfully checked out, simply select “Ok” and you are ready to make modifications to the latest revision of UpStage.

Figure 3 Processing checkout

**Step 4:**

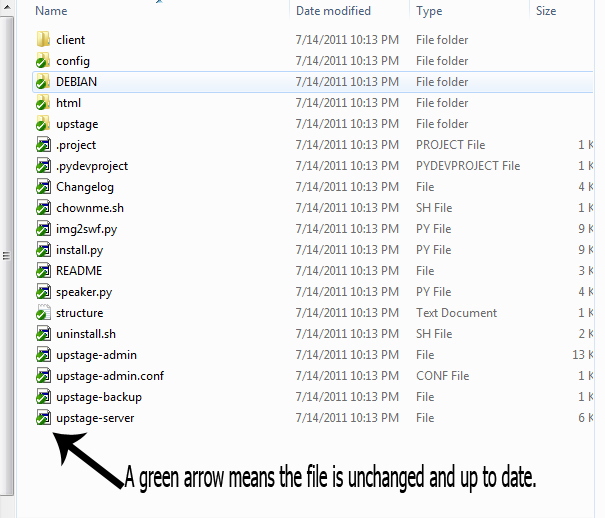
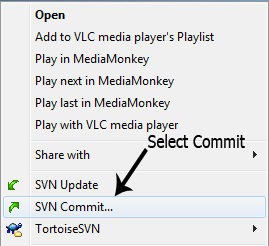
****Browsing the checked out directory will produce a view as outlined in Figure 4. Files or Folders which have a green arrow icon overlay are up to date and unmodified, if a file is modified within the directory a red icon will be shown next to the filename.

Figure 4 Folder View

**Step 5:**

If some modifications have been made, the code can be committed to the repository by right clicking on the directory and selecting “SVN commit” from the context menu.

Figure 5 Context Menu

**Step 6:**

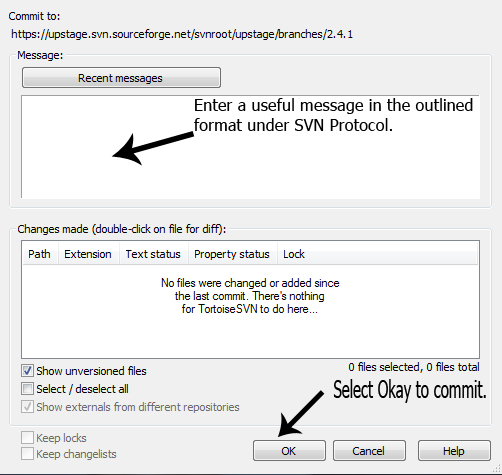
The commit view is fairly self-explanatory the important items to note are the message which needs to be meaningful (Refer to SVN protocol). Any files that have been modified will show up in the changes made area, it will also show files which are not versioned (“unversioned” in SVN terms) in other words new files that need to be added to the repository.

Figure 6 Commit View

Selecting “Ok” will commit the changes to the repository. Please read the SVN Protocol prior to committing as there are important rules to note.

More detailed and advanced information can be obtained at:

<http://sourceforge.net/apps/trac/upstage/wiki/UsingSubversion>

# Git Commit Protocol

A commit should contain related changes only.

Do not commit several changes in one commit.

The following details should also be checked prior to a commit (Consider this a checklist):

* Ensure that any changes made to the code are tested and ensure the code compiles and runs.
* Each commit should contain a newly compiled classes.swf (Refer to UpStage install instructions on how to do this) file to ensure the latest client is always available.
* The commit message should a brief description of changes made in present tense.

*git commit -m “Fix login issue”*

## Versioning

When releasing a new version on GitHub, use the following example command to tag a commit with its hash beginning with ‘df23ad’ (get a commit’s hash using *git log*).

*git tag v3.1.1 df23ad*

*git push origin --tag*

The current versioning system uses 3 digits for example 2.x.x. The third digit represents small fixes such as bugs at this point it is being incremented every 8 bugs fixed. The second digit is used to represent features added. This digit is incremented when at least 4 new features have been added.

The primary digit represents major releases this will represent major changes such as going from unstable to stable.

# Ant Build Automation

**For Development**

You can use ant to compile, build, run UpStage without installing it on the system. You need to install mtasc and swfmill to compile UpStage client.

**Important:** Make sure the following lines in server/src/upstage/config.py are commented and uncommented as appropriate.

*# Uncomment if installed using deb pkg*

*# IMG2SWF\_SCRIPT = '/usr/local/bin/img2swf.py'*

*# Comment below at AUT*

IMG2SWF\_SCRIPT **=** './img2swf.py'

**Ant Targets:**

* *ant clean-start*   
  to build a new instance of UpStage and run it as a background process
* *ant start*   
  to start the built instance of UpStage as a background process
* *ant stop*   
  to stop the UpStage server instance running in the background
* *ant build*   
  to build UpStage, and build/upstage-server.sh to run the server manually on default ports
* *ant run*   
  to build and run automatically
* *ant clean*   
  to clean build and temp directories
* *ant compile-swf*   
  to compile client swf files

# Installing UpStage

A server manual document is on SVN document repository, or alternatively you can access it on GitHub wiki page: <https://github.com/AUTUpStageTeam/upstage2013/wiki/Server-Manual>.

This following section will guide you through the process of install UpStage dependencies.

Once we have satisfied all the dependencies for UpStage we need to fetch the code base to install.

Checkout the latest revision of upstage from:

[<https://github.com/AUTUpStageTeam/UpStage2013>/](https://upstage.svn.sourceforge.net/svnroot/upstage/branches/)

**Important:** Make sure the following lines in server/src/upstage/config.py are commented and uncommented as appropriate.

*# Uncomment if installed using deb pkg*

*IMG2SWF\_SCRIPT = '/usr/local/bin/img2swf.py'*

*# Comment below at AUT*

*# IMG2SWF\_SCRIPT* ***=*** *'./img2swf.py'*

Open a terminal and cd to the newly checked out directory.

*sudo python install.py*

It should be noted that UpStage can be run without any real installation. In that the server can be started using one very simple command, without having to actually do any installation. This procedure would most likely be suitable for developers in that managing, updating and modifying the server does not require any special procedures.

## Dependencies

These are packages that need to be installed prior to installing UpStage.

* espeak
* festival
* gif2png
* libgif4
* netpbm
* python-twisted <= 8.1.0-4
* python <= 2.5.2
* python-pymad

The following packages may not be available in the Debian repositories but are required by the application.

* lame >= 3.97-0.0
* libgdbmg >= 1.7.3-28
* rsynth >= 2.0-6
* mbrola >= 3.01h-6
* swftools >= 0.9.0-0ubuntu1

If that is the case then download the dependency-debs.tar.gz from the AUT online UpStage wiki under DEV\_FILES. The package contains the following Debian packages:

* lame\_3.97-0.0\_i386.deb
* libgdbmg1\_1.7.3-28\_i386.deb
* mbrola\_3.01h-6\_i386.deb
* rsynth\_2.0-6\_i386.deb
* swftools\_0.9.0-0ubuntu1\_i386.deb

**As root run:**

*apt-get install espeak festival gif2png libgif4 netpbm python-twisted python-pymad*

Extract the dependency-debs archive file. This will create either a new folder or will have a number of .deb files located in the current folder. To install these files change directory to the dependency folder and run as root:

*sudo dpkg -i \*.deb*

**NOTE**. If you did not install the dependencies using the deb files. You will need to install: swftools, lame, libgdbmg, mbrola and rsynth using your distributions package manager.

## Quick start a server daemon

UpStage does not really need to be installed; in fact it can merely be run as a daemon from the current directory. For the purposes of these instructions let’s assume the folder containing the recently checked out source code is called upstage.

To start a daemon server follow the steps:

*cd upstage*

*sudo chmod +x upstage-server*

*sudo ./upstage-server –w 8081 –p 7230*

Essentially running the above commands creates the server with the root directory of the server being in the current folder.

To kill the daemons execute:

*sudo ./upstage-server –k*

If you are unsure whether the server has been stopped or not, the following command can be used which outputs nothing if the server has been stopped:

*sudo lsof –I :8081*

The above command assumes the server was started on port 8081, change the port accordingly. If for some reason “./upstage-server –k” did not kill the server process it can be killed manually using the PID(process id). Simply type:

*sudo kill -9 <the process number obtained from the lsof command>*

## Installation using Debian package

UpStage comes with a precompiled client, however if you want to recompile the client you need to have flex compiler version 2. Refer to the section on compiling client.

UpStage can be installed from source using the install.py script however it is recommended to build a Debian package and install using it instead.

**Install the following build dependencies:**

* build-essential
* autoconf automake
* autotools-dev
* dh-make
* debhelper
* devscripts
* fakeroot
* xutils
* lintian
* pbuilder

**Open up a terminal and run:**

*sudo apt-get install build-essential autoconf automake autotools-dev dh-make debhelper devscripts fakeroot xutils lintian pbuilder*

Once the above build tools are installed make sure you are in the root directory of the UpStage folder that contains the file install.py and execute in terminal.

**Create a deb package using:**

*python install.py deb*

**To install the debian package:**

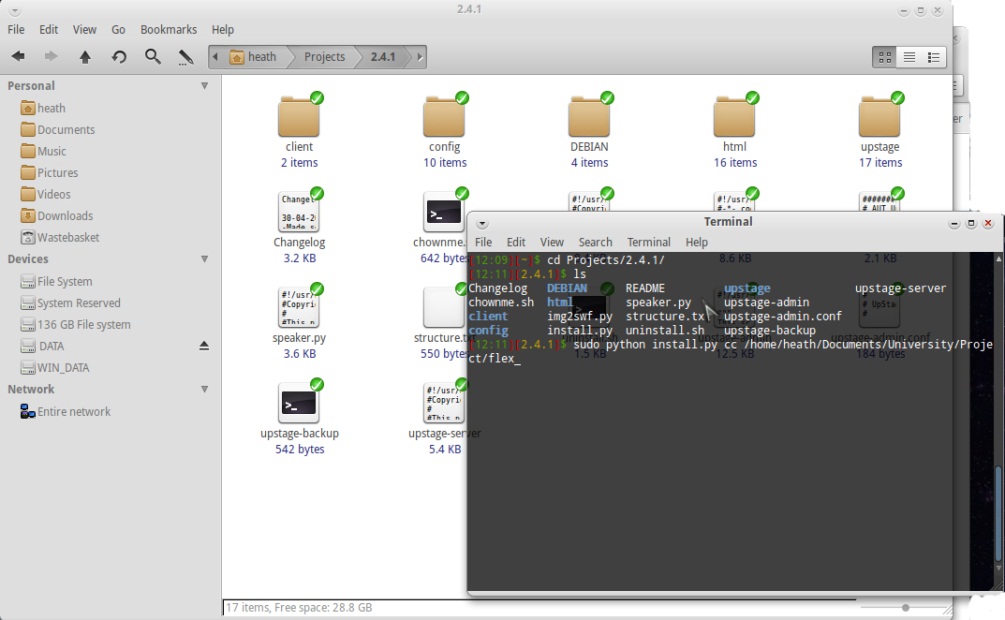
*sudo dpkg –i \*.deb*

The Debian package is a simple way of distributing and removing of the application.

**To uninstall UpStage run:**

*sudo apt-get remove upstage*

## Compiling ActionScript 3 Client

The current unstable development branch of UpStage uses an ActionScript 3 client. This differs from the stable version in that it has been converted from ActionScript 2. In order to compile the client one requires the flex2 compiler which is available on the wiki and some basic terminal knowledge.

Firstly we will cover compiling under Linux based distributions then under windows.

Change directory to the UpStage root folder.

*cd <name folder>*

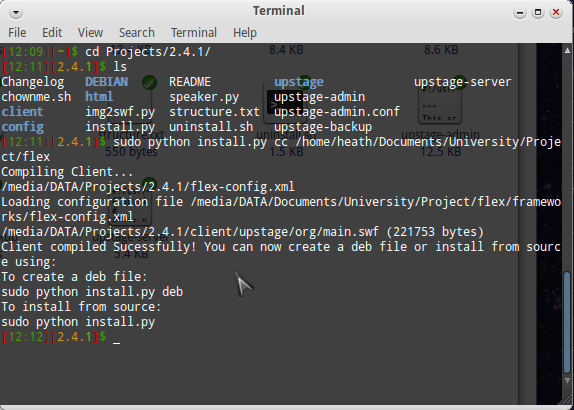
Ensure that you have extracted the flex compiler and note the location of the extracted directory. The install.py script which is included with UpStage handles compiling the client, however it requires a path to the flex directory that had been extracted earlier.

To compile the client use the following command:

*sudo python install.py cc <the path to the flex folder>*

For example:

*sudo python install.py cc /home/upstage-admin/flex*

If the client has been successfully compiled the output will be the same as the below screenshot.

The newly compiled *classes.swf* file will be located under “html/swf/” the name of which will be “*classes.swf*”. UpStage can now be installed with the latest client or if you simply want to add the new client to an existing server, it can be FTP to the server and used immediately. It is recommended to use the install.py compile function always as this has been put in place to make compiling the client simple.

# Voices

# How Voices work

Voices are a key component of UpStage, as such having all the voices available is needed. The package “mbrola” which had been installed previously comes with some voices namely the voices prepended with “e\_”, however voices with “emb\_” and “cmu\_” are not included they need to be installed using other packages. There are Debian packages available however some of them appear to install the voices to incorrect directories. As such we have downloaded all the “emb\_” voices as source code and added them to 4 archives placed on the wiki.

**The following is a basic structure of a voice script:**

/usr/bin/timeout 15 /usr/bin/espeak –k27 -v en/en-n --stdin –w $1.wav ;

***The $ symbol signifies a variable, these examples use a switch.***

S/usr/bin/timeout 15 lame-S –quiet –m m –s 22.05 –preset phone $1.wav ;

***This is writing the bytes to the temporary file $1.wav then passes to lame in which encodes it to .mp3.***

rm $1.wav

***This final line removes the temporary file.***

**E-Speak Directory Structure:**

**Main directory:**

/usr/share/espeak-data

**Mbrola Directory:**

/usr/share/espeak-data/mbrola

/usr/share/espeak-data/mbrola\_ph **- Contains the binary files used for transcoding.**

/usr/share/espeak-data/voices **- Contains all the mbrola voices that are available to espeak.**

/usr/share/espeak-data/voices/mb **-Contains the pointers to voice files.**

# Installing Voices

Each archive is 60mb+ in size. To install the voices download them this can be done using a web browser or if using an SSH session using wget.

Once the 4 zip files have been downloaded, extract them and open up a terminal. Switch to the directory containing all the voice files (Note the files themselves not the directories).

The voice files need to be extracted to “/usr/share/mbrola”, to do this run:

*sudo cp \* /usr/share/mbrola/*

That covers the “emb\_” voices; however there are additional voices that need to be installed thankfully these are located within the Debian repositories.

**Install the EMB Voices:**

*sudo apt-get install  festvox-czech-ph festvox-hi-nsk festvox-italp16k  festvox-itapc16k  festvox-kdlpc16k  festvox-mr-nsk  festvox-suopuhe-common  festvox-suopuhe-lj festvox-suopuhe-mv  festvox-te-nsk*

That will cover about 80% of the voices used by UpStage, the remaining 20% though are more difficult to install and are about 100mb each.

All the “cmu\_” voices can be located <http://festvox.org/cmu_arctic/> as mentioned previously the archives are 100mb each so they total about 600mb for the 6 archives.

**First thing first, install the required dependencies**

sudo apt-get install festival festlex-cmu festlex-poslex festlex-oald libestools1.2 unzip

**Install the voices using this code snippet**

cd /usr/share/festival/voices/english/

sudo wget -c http://www.speech.cs.cmu.edu/cmu\_arctic/packed/cmu\_us\_clb\_arctic-0.95-release.tar.bz2

sudo tar jxf cmu\_us\_clb\_arctic-0.95-release.tar.bz2

sudo ln -s cmu\_us\_clb\_arctic cmu\_us\_clb\_arctic\_clunits

sudo cp /etc/festival.scm /etc/festival.scm.backup

sudo echo "(set! voice\_default 'voice\_cmu\_us\_clb\_arctic\_clunits)" >> /etc/festival.scm

**To set a new default voice simply comment out the previous one and replace it with the new one.**

;;(set! voice\_default 'voice\_cmu\_us\_clb\_arctic\_clunits)

The above procedure is a simple method of testing festival. Below is a more complex method however will provide a number of high quality voices for use with UpStage.

## Installing Enhanced MBROLA Voices

**Download everything needed for the English voices into a temporary directory (total download size is approximately 20 megabytes).**

mkdir mbrola\_tmp

cd mbrola\_tmp/

wget http://tcts.fpms.ac.be/synthesis/mbrola/bin/pclinux/mbrola3.0.1h\_i386.deb

wget -c http://tcts.fpms.ac.be/synthesis/mbrola/dba/us1/us1-980512.zip

wget -c http://tcts.fpms.ac.be/synthesis/mbrola/dba/us2/us2-980812.zip

wget -c http://tcts.fpms.ac.be/synthesis/mbrola/dba/us3/us3-990208.zip

wget -c http://www.festvox.org/packed/festival/latest/festvox\_us1.tar.gz

wget -c http://www.festvox.org/packed/festival/latest/festvox\_us2.tar.gz

wget -c http://www.festvox.org/packed/festival/latest/festvox\_us3.tar.gz

**Install mbrola package (if not already installed).**

sudo dpkg -i mbrola3.0.1h\_i386.deb

**Unpack voices and binary wrappers.**

unzip -x us1-980512.zip

unzip -x us2-980812.zip

unzip -x us3-990208.zip

tar xvf festvox\_us1.tar.gz

tar xvf festvox\_us2.tar.gz

tar xvf festvox\_us3.tar.gz

**Create the directories required for the voices to be installed to.**

sudo mkdir -p /usr/share/festival/voices/english/us1\_mbrola/

sudo mkdir -p /usr/share/festival/voices/english/us2\_mbrola/

sudo mkdir -p /usr/share/festival/voices/english/us3\_mbrola/

**Install the voices and wrappers to the newly created directories.**

sudo mv us1 /usr/share/festival/voices/english/us1\_mbrola/

sudo mv us2 /usr/share/festival/voices/english/us2\_mbrola/

sudo mv us3 /usr/share/festival/voices/english/us3\_mbrola/

sudo mv festival/lib/voices/english/us1\_mbrola/\* /usr/share/festival/voices/english/us1\_mbrola/

sudo mv festival/lib/voices/english/us2\_mbrola/\* /usr/share/festival/voices/english/us2\_mbrola/

sudo mv festival/lib/voices/english/us3\_mbrola/\* /usr/share/festival/voices/english/us3\_mbrola/

**If you want to clean up the temporary directories (We recommend leaving the directory present just in case) simply run the below commands.**

cd ../

rm -rf mbrola\_tmp/

## CMU Arctic Voices

**Download the CMU voices to a temporary directory once again. Note each archive is about 100mb, so it is suggested to run each wget command by itself.**

mkdir cmu\_tmp

cd cmu\_tmp/

wget -c http://www.speech.cs.cmu.edu/cmu\_arctic/packed/cmu\_us\_awb\_arctic-0.90-release.tar.bz2

wget -c http://www.speech.cs.cmu.edu/cmu\_arctic/packed/cmu\_us\_bdl\_arctic-0.95-release.tar.bz2

wget -c http://www.speech.cs.cmu.edu/cmu\_arctic/packed/cmu\_us\_clb\_arctic-0.95-release.tar.bz2

wget -c http://www.speech.cs.cmu.edu/cmu\_arctic/packed/cmu\_us\_jmk\_arctic-0.95-release.tar.bz2

wget -c http://www.speech.cs.cmu.edu/cmu\_arctic/packed/cmu\_us\_rms\_arctic-0.95-release.tar.bz2

wget -c http://www.speech.cs.cmu.edu/cmu\_arctic/packed/cmu\_us\_slt\_arctic-0.95-release.tar.bz2

**Simply unpack the voices using the below bash script.**

for t in `ls cmu\_\*` ; do tar xf $t ; done

rm \*.bz2

**Install the voices. The below commands creates the required directory for the voices, then moves all files to the newly created directory.**

sudo mkdir -p /usr/share/festival/voices/english/

sudo mv \* /usr/share/festival/voices/english/

**Festival requires them to have slightly different directory names, so rename them.**

for d in `ls /usr/share/festival/voices/english` ; do

if [[ "$d" =~ "cmu\_us\_" ]] ; then

sudo mv "/usr/share/festival/voices/english/${d}" "/usr/share/festival/voices/english/${d}\_clunits" fi ; done

It should be noted that the voice directories themselves do not need to be present within “/usr/share/festival/voices/english” instead they can be linked to that directory from anywhere on the system using:

*ln –s <path to voice folder eg. cmu\_us\_jmk\_arctic> <location of link>*

For example:

*ln –s /home/upstage-admin/cmu\_us\_jmk\_arctic cmu\_us\_jmk\_arctic\_clunits*

## Testing Festival

**A festival console can be accessed through the terminal by typing**

*festival*

**This will present a > prompt, to use festival within the prompt first see if there are any voices available this is done using**

*(voice.list)*

**Set a voice**

*(voice\_en1\_mbrola)*

**If there are some voices linked to the festival console they can be used by typing**

*(SayText “Hello”)*

**One can also call text from a text file using the following command**

*(tts “FILENAME” nil)*

**Exit**

*(exit)*

# Managing UpStage

The below commands should all be run as root. So either login as root using ‘*su*’ or append all commands with ‘*sudo’*.

**Commands for UpStage-server**

**Start a server daemon on port 8081 with the swf port on 7230**

*upstage-server –w 8081 –p 7230*

**Kill a server daemon**

*upstage-server –k*

**Commands for upstage-admin**

**Create a new server process with the name test**

*upstage-admin create test*

**List all running server processes**

*upstage-admin ls*

**Start a server process**

*upstage-admin start <name>*

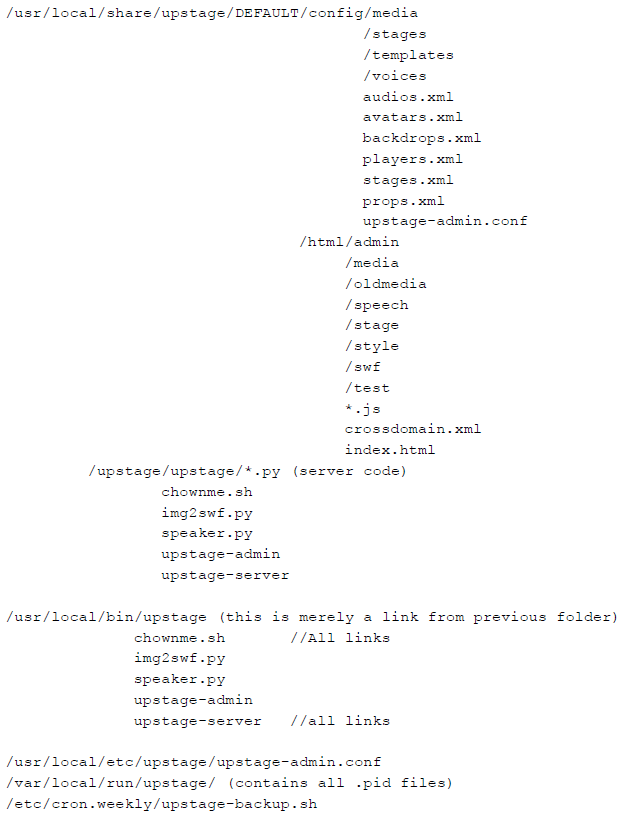
**Stop a server process**

*upstage-admin stop <name>*

**Restart a server process**

*upstage-admin restart <name>*

# UpStage Directory Structure



# Introduction to Linux

It is important to note that under Linux operating systems there are two main types of users, namely super users (root) and regular users. In order to install applications and perform tasks such as killing root processes require root access. In Debian one can login as root in a terminal using the command “*su*” or “*sudo su*” the root password will need to be entered. Once you are logged in as root, everything within the system can be altered, so it is important to be careful when running commands, especially removing files and/or killing processes.

Logging in as root means that there will be no ‘permission denied’ messages when running any commands. If you prefer it is possible to prefix commands with “*sudo*” which will run a particular command with root privileges without actually logging in as root this is the “better” approach however when running a lot of commands in a given time frame it is certainly easier to simply login as root.

## Linux File System Basics

For the purposes of this manual we will stay within in the Debian realm, that being said the file system does not change that much from one distribution to another. The Linux file system is broken into “/” directories, the most common of which are /home which contains all files for a given user and /root which is the home for the root user. I won’t go into detail about /usr, /bin, /etc however they will be referred to. Most work done by a local user need only be done in the users /home directory for example if a user with the name “upstage-admin” existed on the system then their home directory would be located in “/home/upstage-admin/” Accessing a directory within the file system can be done using Nautilus which is a file manager packaged with Debian. However In most cases it is easier and more efficient to simply navigate the file system using the terminal.

To navigate to a particular directory such as “/home/upstage-admin” one can run “cd /home/upstage-admin” this will navigate to the home directory of upstage-admin from anywhere in the filesystem.

An important note is that when using the “cd” command, using a “/” tells the command that the file is within the root directory so for example using “cd /home”

would change to “/home” however using “cd home” would look for a directory within the current directory.

## Installing Applications

Installing applications within a Linux environment is very simple; most distributions use a package manager which provides a repository which holds application packages. In most cases these applications are precompiled and only require a simple command to install or depending on the distribution a graphical package manager may be provided. There are a number of different package managers for the purpose of this document; we will be covering the Debian Package Manager. Debian does have a graphical package manager named Synaptic, however in most cases faster to use “aptitude” through a terminal. Below are some of the simple commands used to install and remove packages within Debian.

**Note: Installing applications through a package manager requires elevated privileges (root access) so ensure that you run the command as root or prepend commands with “*sudo*”**

**Update:**

*sudo apt-get update*

**Search for a package named upstage:**

*aptitude search upstage*

**Install a package named ssh:**

*sudo apt-get install ssh*

**Wildcards can also be used for example:**

*sudo apt-get install ssh\**

The above command will install all packages that start with SSH.

**Remove an installed package (in this case ssh):**

*sudo apt-get remove ssh*

**Remove packages that are no longer required:**

*sudo apt-get autoremove*

## Simple Bash Commands

|  |  |
| --- | --- |
| Command | Description |
| adduser | Add a user to the system |
| addgroup | Add a group to the system |
| alias | Create an alias |
| apropos | Search Help manual pages (man -k) |
| apt-get | Search for and install software packages (Debian/Ubuntu) |
| aptitude | Search for and install software packages (Debian/Ubuntu) |
| aspell | Spell Checker |
| awk | Find and Replace text database sort/validate/index |
| basename | Strip directory and suffix from filenames |
| bash | GNU Bourne-Again SHell |
| bc | Arbitrary precision calculator language |
| bg | Send to background |
| break | Exit from a loop |
| builtin | Run a shell builtin |
| bzip2 | Compress or decompress named file(s) |
| cal | Display a calendar |
| case | Conditionally perform a command |
| cat | Concatenate and print (display) the contents of file(s) |
| cd | Change Directory |
| cfdisk | Partition table manipulator for Linux |
| chgrp | Change group ownership |
| chmod | Change access permissions |
| chown | Change file owner and group |
| chroot | Run a command with a different root directory |
| chkconfig | System services (runlevel) |
| cksum | Print CRC checksum and byte counts |
| clear | Clear terminal screen |
| cmp | Compare two files |
| comm | Compare two sorted files line by line |
| command | Run a command - ignoring shell functions |
| continue | Resume the next iteration of a loop |
| cp | Copy one or more files to another location |
| cron | Daemon to execute scheduled commands |
| crontab | Schedule a command to run at a later time |
| csplit | Split a file into context-determined pieces |
| cut | Divide a file into several parts |
| date | Display or change the date & time |
| dc | Desk Calculator |
| dd | Convert and copy a file write disk headers boot records |
| ddrescue | Data recovery tool |
| declare | Declare variables and give them attributes |
| df | Display free disk space |
| diff | Display the differences between two files |
| diff3 | Show differences among three files |
| dig | DNS lookup |
| dir | Briefly list directory contents |
| dircolors | Colour setup for `ls' |
| dirname | Convert a full pathname to just a path |
| dirs | Display list of remembered directories |
| dmesg | Print kernel & driver messages |
| du | Estimate file space usage |
| echo | Display message on screen |
| egrep | Search file(s) for lines that match an extended expression |
| eject | Eject removable media |
| enable | Enable and disable builtin shell commands |
| env | Environment variables |
| ethtool | Ethernet card settings |
| eval | Evaluate several commands/arguments |
| exec | Execute a command |
| exit | Exit the shell |
| expect | Automate arbitrary applications accessed over a terminal |
| expand | Convert tabs to spaces |
| export | Set an environment variable |
| expr | Evaluate expressions |
| false | Do nothing unsuccessfully |
| fdformat | Low-level format a floppy disk |
| fdisk | Partition table manipulator for Linux |
| fg | Send job to foreground |
| fgrep | Search file(s) for lines that match a fixed string |
| file | Determine file type |
| find | Search for files that meet a desired criteria |
| fmt | Reformat paragraph text |
| fold | Wrap text to fit a specified width. |
| for | Expand words and execute commands |
| format | Format disks or tapes |
| free | Display memory usage |
| fsck | File system consistency check and repair |
| ftp | File Transfer Protocol |
| function | Define Function Macros |
| fuser | Identify/kill the process that is accessing a file |
| gawk | Find and Replace text within file(s) |
| getopts | Parse positional parameters |
| grep | Search file(s) for lines that match a given pattern |
| groups | Print group names a user is in |
| gzip | Compress or decompress named file(s) |
| hash | Remember the full pathname of a name argument |
| head | Output the first part of file(s) |
| help | Display help for a built-in command |
| history | Command History |
| hostname | Print or set system name |
| iconv | Convert the character set of a file |
| id | Print user and group id's |
| if | Conditionally perform a command |
| ifconfig | Configure a network interface |
| ifdown | Stop a network interface |
| ifup | Start a network interface up |
| import | Capture an X server screen and save the image to file |
| install | Copy files and set attributes |
| jobs | List active jobs |
| join | Join lines on a common field |
| kill | Stop a process from running |
| killall | Kill processes by name |
| less | Display output one screen at a time |
| let | Perform arithmetic on shell variables |
| ln | Make links between files |
| local | Create variables |
| locate | Find files |
| logname | Print current login name |
| logout | Exit a login shell |
| look | Display lines beginning with a given string |
| lpc | Line printer control program |
| lpr | Off line print |
| lprint | Print a file |
| lprintd | Abort a print job |
| lprintq | List the print queue |
| lprm | Remove jobs from the print queue |
| ls | List information about file(s) |
| lsof | List open files |
| make | Recompile a group of programs |
| man | Help manual |
| mkdir | Create new folder(s) |
| mkfifo | Make FIFOs (named pipes) |
| mkisofs | Create an hybrid ISO9660/JOLIET/HFS filesystem |
| mknod | Make block or character special files |
| more | Display output one screen at a time |
| mount | Mount a file system |
| mtools | Manipulate MS-DOS files |
| mtr | Network diagnostics (traceroute/ping) |
| mv | Move or rename files or directories |
| mmv | Mass Move and rename (files) |
| netstat | Networking information |
| nice | Set the priority of a command or job |
| nl | Number lines and write files |
| nohup | Run a command immune to hangups |
| notify-send | Send desktop notifications |
| nslookup | Query Internet name servers interactively |
| open | Open a file in its default application |
| op | Operator access |
| passwd | Modify a user password |
| paste | Merge lines of files |
| pathchk | Check file name portability |
| ping | Test a network connection |
| pkill | Stop processes from running |
| popd | Restore the previous value of the current directory |
| pr | Prepare files for printing |
| printcap | Printer capability database |
| printenv | Print environment variables |
| printf | Format and print data |
| ps | Process status |
| pushd | Save and then change the current directory |
| pwd | Print Working Directory |
| quota | Display disk usage and limits |
| quotacheck | Scan a file system for disk usage |
| quotactl | Set disk quotas |
| ram | ram disk device |
| rcp | Copy files between two machines |
| read | Read a line from standard input • |
| readarray | Read from stdin into an array variable • |
| readonly | Mark variables/functions as readonly |
| reboot | Reboot the system |
| rename | Rename files |
| renice | Alter priority of running processes |
| remsync | Synchronize remote files via email |
| return | Exit a shell function |
| rev | Reverse lines of a file |
| rm | Remove files |
| rmdir | Remove folder(s) |
| rsync | Remote file copy (Synchronize file trees) |
| screen | Multiplex terminal run remote shells via ssh |
| scp | Secure copy (remote file copy) |
| sdiff | Merge two files interactively |
| sed | Stream Editor |
| select | Accept keyboard input |
| seq | Print numeric sequences |
| set | Manipulate shell variables and functions |
| sftp | Secure File Transfer Program |
| shift | Shift positional parameters |
| shopt | Shell Options |
| shutdown | Shutdown or restart linux |
| sleep | Delay for a specified time |
| slocate | Find files |
| sort | Sort text files |
| source | Run commands from a file `.' |
| split | Split a file into fixed-size pieces |
| ssh | Secure Shell client (remote login program) |
| strace | Trace system calls and signals |
| su | Substitute user identity |
| sudo | Execute a command as another user |
| sum | Print a checksum for a file |
| suspend | Suspend execution of this shell |
| symlink | Make a new name for a file |
| sync | Synchronize data on disk with memory |
| tail | Output the last part of files |
| tar | Tape Archiver |
| tee | Redirect output to multiple files |
| test | Evaluate a conditional expression |
| time | Measure Program running time |
| times | User and system times |
| touch | Change file timestamps |
| top | List processes running on the system |
| traceroute | Trace Route to Host |
| trap | Run a command when a signal is set(bourne) |
| tr | Translate squeeze and/or delete characters |
| true | Do nothing successfully |
| tsort | Topological sort |
| tty | Print filename of terminal on stdin |
| type | Describe a command |
| ulimit | Limit user resources |
| umask | Users file creation mask |
| umount | Unmount a device |
| unalias | Remove an alias |
| uname | Print system information |
| unexpand | Convert spaces to tabs |
| uniq | Uniquify files |
| units | Convert units from one scale to another |
| unset | Remove variable or function names |
| unshar | Unpack shell archive scripts |
| until | Execute commands (until error) |
| useradd | Create new user account |
| usermod | Modify user account |
| users | List users currently logged in |
| uuencode | Encode a binary file |
| uudecode | Decode a file created by uuencode |
| v | Verbosely list directory contents (`ls -l -b') |
| vdir | Verbosely list directory contents (`ls -l -b') |
| vi | Text Editor |
| vmstat | Report virtual memory statistics |
| watch | Execute/display a program periodically |
| wc | Print byte word and line counts |
| whereis | Search the user's $path man pages and source files for a program |
| which | Search the user's $path for a program file |
| while | Execute commands |
| who | Print all usernames currently logged in |
| whoami | Print the current user id and name (`id -un') |
| Wget | Retrieve web pages or files via HTTP HTTPS or FTP |
| write | Send a message to another user |
| xargs | Execute utility passing constructed argument list(s) |
| xdg-open | Open a file or URL in the user's preferred application. |
| yes | Print a string until interrupted |
| . | Run a command script in the current shell |
| ### | Comment / Remark |

## More complex Commands

|  |  |
| --- | --- |
| Command | Description |
| kill `lsof -i :8080 | awk '{print $2}' | egrep "[0-9]{1,}" -o` | Kills a process based on the listening port. |
| for pid in $(ps -ef | grep defunct | grep -v grep | awk '{ print $3 }'); do kill $pid; done | Kills all defunct (Zombie processes) on the system. |
| kill `ps –A | grep upstage | awk ‘{ print $1 }’` | Kills the upstage process on the server. |
| tail –f somefile | Watches the file for changes and prints out to the terminal the text changes made to the file. Useful for monitoring the UpStage server log files. |
| for pid in $( ps –A | grep upstage | awk ‘{ print $1 }’);do kill $pid; done | Kill all running upstage processes on the server. |
|  |  |
|  |  |

## Basic Command Usage

**Print current working directory**

*pwd*

**Change working directory to desktop**

*cd Desktop*

**Go back one directory**

*cd ..*

**Go back to the current user’s home directory**

*cd ~/*

**Remove directory named “upstage”**

*rm –rf upstage*

**Copy file to the directory upstage**

*cp somefile.txt upstage*

**Move file to the upstage directory**

*mv somefile.txt upstage/somefile.txt*

**Give all permissions on a file**

*chmod 777 somefile.py*

**Make file executable**

*chmod +x somefile.py*

**List files in directory**

*ls –l*

**View all processes with upstage in their name**

*ps –A | grep upstage*

**Output contents of a file to terminal**

*cat somefile.txt*

**Execute a Python file**

*python somefile.py*

**Execute file in current directory (File needs to be marked as executable)**

*./someexecutablefile*

**View processes listening on port 8081**

*lsof –i :8081*

**Create a soft link with the name “link” to a directory “/home/upstage-admin/upstage:**

*ln –s /home/upstage-admin/upstage/ link*

# SSH (Secure Shell)

SSH is a secure method to connect to a remote Linux desktop using a terminal session. This allows a user to connect to the UpStage server and run commands as if they were sitting on the server. Of course SSH does not provide a graphical environment so everything needs to be done using the terminal command line.

The SSH commands remain the same whether using Windows or Linux however Windows requires a 3rd party application to be installed called [Putty](http://www.chiark.greenend.org.uk/~sgtatham/putty/) which is a cross platform SSH client, if using a Linux distribution SSH will be available in the repositories and will most likely need to be installed.

**Install SSH:**

*sudo apt-get install ssh*

**Once SSH is installed an SSH connection can be established using:**

*ssh user@address*

**For example:**

*ssh upstage-admin@156.62.62.18*

**If a port other than 22 has been specified use:**

*ssh –p portnumber user@address*

**To exit an SSH session type:**

*exit*

# Development Environment

Windows

<http://notepad-plus-plus.org/>

All

<http://www.sublimetext.com/2>

# Debugging

**Terminal:**

Starting UpStage using “*./upstage-server –n –p 7230 –w 8083”* will output logs to stdout essentially providing debugging through a terminal not a log file.

**File:**

The alternative method is to use tail which monitors a file for changes and outputs to the terminal. For example to monitor the UpStage log file “*tail –f <locationoflogfile>/upstage.log”*

The file can also be opened in a text editor however doing so locks the file for write access.

**Eclipse:**

In order to use Eclipse as a debugger this would be possible if Debian 5 was used as the development environments operating system. This would provide similar use to that of Netbeans with java, resulting in UpStage being run locally and completely independent of other developers work. The current team gets around this by using multiple isolated servers on the local server, each running on unique ports.

# Coding Standards

Please read the coding standards PDF document on the SVN document repository, or <https://github.com/AUTUpStageTeam/upstage2013/wiki/Coding-Standards>

# PyLint

PyLint analyzes Python source code looking for bugs and signs of poor quality. It is a python tool that checks if a module satisfies a coding standard which is fully configurable. The module is able to do among many things check the line length of functions, check if variable names are well-formed according to your coding standard, or checking if declared interfaces are truly implemented. The big advantage with Pylint over other applications is that it is highly configurable, customizable, and one can easily write a small plugin to add a personal feature.

PyLint can be obtained at: <http://www.logilab.org/857> or through the package manager on most (all) Linux distributions.

The User Manual is available at: <http://www.logilab.org/card/pylint_manual>

Using PyLint

**Running PyLint against a given Python file is as simple as firing up a terminal and running:**

*pylint <python source file>*

The output from the above command can look complex at first however a very detailed tutorial and the interpretation of the output is available at: <http://www.logilab.org/card/pylint_tutorial>

Pylint can also be used with Eclipse which will show errors while coding using highlighting. The instructions to install Pylint for usage with Eclipse can be found at:

<http://code.google.com/p/soc/wiki/SetUpPylint>

# Pep8

Pep8 is a tool used to check Python code against some of the style conventions outlined in the standard Python coding standards available at:

<http://www.python.org/dev/peps/pep-0008/>

The tool can be easily downloaded from <http://www.python.org/dev/peps/pep-0008/> along with installation instructions and usage. However running the Pep8 module against a python file is merely a matter of opening up a terminal and executing:

*pep8 <python file to check>*

**For example running the module against optparse.py results in the following output:**

*$ pep8 optparse.py*

*optparse.py:69:11: E401 multiple imports on one line*

*optparse.py:77:1: E302 expected 2 blank lines, found 1*

*optparse.py:88:5: E301 expected 1 blank line, found 0*

*optparse.py:222:34: W602 deprecated form of raising exception*

*optparse.py:347:31: E211 whitespace before '('*

**All the available information for usage can be obtained using:**

*pep8 –h*

# Language Basics

## Python Basics

The Python interpreter operates somewhat like the UNIX shell: when called with standard input connected to a TTY device, it reads and executes commands interactively; when called with a file name argument or with a file as standard input, it reads and executes a script from that file.

The interpreter provides a method of testing commands prior to adding them to an application. It can be accessed by typing:

*python*

This can be done in a terminal under any Linux distribution and the same can be done under a windows machine; which has the python modules installed and setup.

**Python documentation is available here** <http://docs.python.org/release/2.6.7/library/index.html>

**A very good Python tutorial is available here** <http://docs.python.org/tutorial/index.html>

## ActionScript Basics

**ActionScript documentation** <http://www.adobe.com/support/documentation/en/flash/>

# AUT UpStage Server (WT501)

## Setting up vSFTP

In order to make development easier the AUT server runs an FTP server so that files can be uploaded and download to and from the server. Modified files can be uploaded to the server replacing a current file and the changes can be tested immediately. Setting up and FTP server is not that complex however a tutorial has been included below.

**Installation:**

*sudo apt-get install vsftpd*

**Modify /etc/hosts.allow in order to allow connections to vsftpd using a text editor (nano, gedit, etc.):**

*# Allow all connections*

*vsftpd: ALL*

*# To Restrict to a certain IP range*

*#vsftpd: 156.62.62.0/255.255.255.0*

**To start the server run execute as root:**

*# /etc/init.d/vsftpd start*

Most of the settings in vsftpd are done by editing the file /etc/vsftpd.conf. The file itself is well-documented, so this section only highlights some important changes you may want to modify.

### Enabling uploading

The WRITE\_ENABLE flag must be set to YES in /etc/vsftpd.conf in order to allow changes to the file system, such as uploading:

*write\_enable=YES*

### Local user logging

One must set the line to /etc/vsftpd.conf to allow users in /etc/passwd to login:

*local\_enable=YES*

### Anonymous logging

The line in /etc/vsftpd.conf controls whether anonymous users can login:

*anonymous\_enable=YES # Allow anonymous login*

*no\_anon\_password=YES # No password is required for an anonymous login*

*anon\_max\_rate=30000 # Maximum transfer rate for an anonymous client in bytes per second*

We chose to use the below method however currently there is no need for security per se therefore using the above method does require less tinkering. If for some reason security is an issue and you want to only allow local users to login then method outlined below is required.

### Chroot jail

One can set up a chroot environment which prevents the user from leaving its home directory. To enable this, add the following lines to /etc/vsftpd.conf:

*chroot\_list\_enable=YES*

*chroot\_list\_file=/etc/vsftpd.chroot\_list*

The chroot\_list\_file variable specifies the file which contains users that are jailed.

For a more restricted environment, one can specify the line:

*chroot\_local\_user=YES*

This will make local users jailed by default; in this case, the file specified by chroot\_list\_file lists users that are **not** in a chroot jail.

The VSFTP man pages can be accessed via <http://vsftpd.beasts.org/vsftpd_conf.html>

To avoid permission errors it is recommended to simply provide all permissions on any files or directories you want available to users.

**This can be done using the following command:**

*chmod 777 filename*

A client is required to connect to the server the most popular FTP client is ***Filezilla*** located [here](http://filezilla-project.org/).

# Flash Tracer (Debugging Flash)

Debugging Actionscript requires the installation of a different flash plugin and in most cases it will overwrite the standard flash plugin. The flash tracer plugin only works with Firefox however the Flash debugger plugin works with all browsers.

**Download Flash Debugger:**

<http://www.adobe.com/support/flashplayer/downloads.html>

As you will notice there are different versions, the one used under linux is:

[Download the Linux Flash Player 10.3 Plugin content debugger](http://download.macromedia.com/pub/flashplayer/updaters/10/flashplayer_10_plugin_debug.tar.gz)

The one used under Windows is:

[Download the Windows Flash Player 10.3 ActiveX control content debugger (for IE)](http://download.macromedia.com/pub/flashplayer/updaters/10/flashplayer_10_ax_debug.exe)

Or

[Download the Windows Flash Player 10.3 Plugin content debugger (for Netscape-compatible browsers)](http://download.macromedia.com/pub/flashplayer/updaters/10/flashplayer_10_plugin_debug.exe)

The above link is for browsers such as Firefox and Google chrome.

Installation under windows is as simple as running the executable however under Linux based systems one needs to first locate the libflashplayer.so file.

### Linux Install:

**Extract the downloaded archive file:**

*tar zxvf <filename>*

**Locate the libflashplayer.so using:**

*sudo find / -name libflashplayer.so*

The output should be similar to the screenshot below.

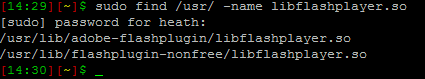


Figure 7: find output libflashplayer.so

Generally speaking on a Debian based system depending on the browser those are the 2 locations which a browser will look for the flash plugin. It is a good idea to back up the current libflashplugin.so in case something goes wrong.

**Back up the current flash plugin:**

*sudo cp <location of libflashplayer.so>/libflashplayer.so <location of libflashplayer.so>/libflashplayer.so.back*

**For example:**

*sudo cp /usr/lib/adobe-flashplugin/libflashplayer.so /usr/lib/adobe-flashplugin/libflashplayer.so.back*

**Copy the extracted libflashplayer.so and overwrite the one stored in for example “/usr/lib/adobe-flashplugin/”:**

*sudo cp libflashplayer.so /usr/lib/adobe-flashplugin/libflashplayer.so*

If you have a web browser open it should have an error message that flash has crashed, if not repeat the above steps for other possible locations of libflashplayer.so. Once flash debug has been installed we can now install the flash tracer plugin.

Flash tracer can be obtained at: <http://www.sephiroth.it/weblog/archives/2006/10/flashtracer_firefox_extensionphp.php>

Installation is as simple as clicking the link and Firefox will install the add-on automatically.

Once this has completed, flash tracer needs to know where the flashlog.txt file is located. Generally under windows the file is located under:

*C:\Documents and Settings\{user}\Application Data\Macromedia\Flash Player\Logs\flashlog.txt*

Pointing flash tracer to the above file path will allow flash tracer to monitor the file for changes. The same can be done under Linux based systems using a different path, namely:

*/home/<user>/.Macromedia/Flash\_Player/Logs/flashlog.txt*

Also we need to add another file under each user’s home directory.

• Windows; C:\Documents and Settings\username\mm.cfg

• Linux; home/<user>/mm.cfg

**The file needs to contain tracing parameters in this case:**

ErrorReportingEnable=1

TraceOutputFileEnable=1

TraceOutputFileName=***<Location of the flashlog.txt>***

MaxWarnings=50

If all is installed correctly trace() messages should be output the flashlog.txt, one can monitor this file using the terminal or the flashtracer plugin.