

Fun with XMMS

(and friends)

Tips and tricks for the unix desktop

Joshua Malone
(*jmalone@ubergeeks.com*)

A Brief Look at the X11 Login Process

Display Managers

Provide graphical login to X

- GDM – gnome display manager
- KDM – KDE display manager
- WDM – Wings (WindowMaker) display manager
- XDM – “Classic” Xlib display manager

Xinit

- File `~/.xinitrc` and `~/.xsession`
- I just `ln -s .xinitrc .xsession`
- Shell script that controls your X startup

```
#!/bin/bash
eval $(ssh-agent)
ssh-add ~/.ssh/id_dsa
bbkeys -w &
blackbox

if [ $? ]; then
    xterm
fi

ssh-agent -k

#start SSH agent to hold keys
#load SSH keys into agent
#accessory for blackbox WM
#start window manager
#Notice the lack of '&'

#if WM exits abnormally,
start
# an emergency xterm
#kill SSH agent when done
```

The Window Manager

- Controls the look and feel of windows around programs
- May be part of a “desktop environment” such as Gnome, KDE or XFCE
- You don't need a full desktop environment – just a window manager is all you need
- Some lists of window managers:
 - <http://xwinman.org/>
 - <http://freshmeat.net/browse/56/>
 - FreeBSD: consult `/usr/ports/x11-wm/`
- Find the look and feel that suits you

Configuring WDM

- /etc/X11/wdm/

- Xservers

```
# Set display to 100 dpi for higher resolutions
:0 local /usr/X11R6/bin/X -nolisten TCP -dpi 100
```

- wdm-config

```
# Make reboot/halt work without logging in
DisplayManager*wdmVerify:           false
DisplayManager*wdmRoot:              false

DisplayManager*wdmLogo:              /path/to/image.xpm
```

“Auto-launch” programs after your window manager starts

Problem: You want to start programs in your `xinitrc` *after* your window manager has started.

Recall `.xinitrc`:

```
#!/bin/bash
eval $(ssh-agent)
ssh-add ~/.ssh/id_dsa
bbkeys -w &
blackbox

if [ $? ]; then
    xterm
fi

ssh-agent -k
```

“Auto-launch” programs after your window manager starts

Problem: You want to start programs in your `xinitrc` *after* your window manager has started.

Solution: Start window manager in the background and do some job management.

- In your `.xinitrc`

```
#!/bin/bash
window_manager &
WM_PID=$!           # $! == PID of last background process
sleep 1             # give WM a chance to start

other_programs &

wait ${WM_PID}      # This keeps xinit from quitting
```

Advanced XMMS Usage

xmmsctrl

Command-line interface to almost *every* XMMS
button

- <http://user.it.uu.se/~adavid/utils/>
- <http://www.xmms.org/plugins.php?details=11>

xmms + mozilla

xmms-web.sh

```
#!/bin/bash
xmmsctrl stop
xmms -e "${1}"
xmmsctrl track last
xmmsctrl play
```

- Open mp3, m3u, etc. files with this script
- Adds the file to your playlist and plays it automatically
- Keeps "history" in your playlist

xmms plugins

- EQU (equ.sourceforge.net)
 - 31-band graphic equalizer
 - Equalizes any media type (not just MP3s)
- Songchange (included by default)
 - Run an arbitrary command whenever a song change occurs
 - Use in conjunction with xosd or other for a “Now Playing” feature

xmms patches

- id3v2 patch
 - <http://dev.gentoo.org/~eradicator/xmms>
- Disk writer effects patch
 - Use effect plugins while writing to wav file
 - http://bugs.xmms.org/show_bug.cgi?id=662
 - <http://bugs.xmms.org/attachment.cgi?id=250&action=view>

What to do with those extra keys...

xbindkeys

Use those stupid media keys on your keyboard.

- Find the keycodes from "special" keys using 'xev'
- Bind those codes to command line you like!
- Use in combination with 'xmmsctrl' to control XMMS, even if the window is hidden or on another virtual desktop

xbindkeys

Example .xbindkeysrc:

```
# Play/pause (Play button)
"xmms -t"
    c:159
# Vol Up (Volume buttons)
"xmmsctrl vol +5%"
    c:158
# Mute (Mute button)
"xmmsctrl vol 10%"
    c:166
# Title popup (media key)
"xmmsctrl print %T | xmessage -file - -timeout 3"
    c:129
# Seek fwd 20s (ctrl + FF)
"xmmsctrl time +20"
    c:162 + m:4
```

Hacking the Terminal

X Shell Behaviour

- Launch terminal emulator (xterm,gnome-terminal, etc.)
- Terminal emulator launches your default shell (bash, etc.)
 - Stored in /etc/passwd
- Shell reads system-wide config files (/etc/profile, /etc/bash.bashrc, etc.)
 - Sets system \$PATH, default prompt, etc.
- Shell reads your user config files (~/.bashrc, etc.)
 - Override system-wide defaults
 - Run any login scripts (check for interactive use!)
- Shell begins interactive use
- You get a prompt

bash startup

- `~/.bashrc`
 - Read & executed when bash starts interactively
 - Not read when used as login shell
- `~/.bash_profile`, `~/.bash_login`, `~/.profile`
 - Read & executed when invoked as a login shell (xterm -ls)
 - Reads them IN THAT ORDER
- `~/.bash_logout`
 - Read & executed when shell exits

Managing bash config files

- Read ~/.bashrc for logins shells, too

```
# ~/.bash_profile
if [ -f ~/.bashrc ]; then
    . ~/.bashrc
fi
```

- The dot above is an abbreviation for 'source'; this loads and runs another bash script.
- Open square bracket is symlink for 'test' program.
- Always check that the file exists (test -f) before sourcing it or shell will die!

mrxvt

Tabbed multi-terminal emulator

- Formerly multi-aterm
- Lightweight “xterm” replacement with tabbed sessions a-la “konsole”
- <http://materm.sourceforge.net/>
- Resource configurable (~/.Xdefaults)
- Supports: scrollbars, background images, keyboard shortcuts, kanji characters, transparency and more

mrxvt titles

- Change title in window manager frame

```
echo -ne "\033]0; <title text> \007"
```

- (Also works in 'xterm')

- Change title in terminal tab

```
echo -ne "\033]61; <title text> \007"
```

Terminal Enhancements

Use the window manager frame title

- Set title to PWD

Add line to your ~/.bashrc:

```
PROMPT_COMMAND='echo -ne "\033]0;${PWD}\007" '  
export PROMPT_COMMAND
```

- Add hostname to prevent “SSH-confusion”

```
PROMPT_COMMAND='echo -ne "\033]0;${USER}@$(hostname -s)\007" '
```

Detect terminal emulator

- Most terminal emulators export a TERM environment variable

```
case ${TERM} in
  xterm*)          # Catches xterm, xterm-color, etc.
    PROMPT_COMMAND='echo -ne "\033]0;XTERM\007"'
    ;;
  rxvt)
    PROMPT_COMMAND='echo -ne "\033]61;MRXVT\007"'
    ;;
esac
```

ssh-agent

A “keyring” for OpenSSH

In your `.xinitrc`

```
eval $(ssh-agent)
```

Load your private keys into your agent:

```
ssh-add ~/.ssh/id_dsa
```

```
ssh-add ~/.ssh/other_key
```

Access your ssh-agent from a console session

Problem: Environment variables set by xinit aren't available when you log in on text console

Solution: Detect a running ssh-agent in your shell

- In your `.xinitrc`

```
eval $(ssh-agent)
{
echo "export SSH_AUTH_SOCKET=${SSH_AUTH_SOCKET}"
echo "export SSH_AGENT_PID=${SSH_AGENT_PID}"
} > ~/.agentinfo
```

- In your `.bashrc`

```
if [ -f ~/.agentinfo ] && [ -z "${SSH_AUTH_SOCKET}" ];
then
    . ~/.agentinfo
fi
```

Useful Scripting Utils

xmessage / gxmessage

- Display GUI choices from a script
- Prints return code on stdout
- Usage:

```
RESULT=$(xmessage -buttons <buttons> "Message")  
case ${RESULT}  
    ...
```

- Button string:

```
-buttons "Label One:value1,Label 2:value2"
```

Useful Scripting Utils

xconsole

- Not just for /dev/console anymore :)
- Graphical 'tail -f' on a file
- Usage:

```
xconsole -file "${FILE_TO_TAIL}" &  
XCONSOLE_PID=$!
```

```
echo "stuff" >> "${FILE_TO_TAIL}"
```

```
kill ${XCONSOLE_PID}
```

- Caution: If the file is re-created (inode changed) xconsole will not display contents of the new file

Useful Scripting Utils

xprompt

- Prompt user for an input string

My tcl/tk gui-utils tools

These are some tools I've written that I use in my bash scripts when I need a GUI interface.

- tkfilechooser: File selection dialog
- tktail: Graphical tail similar to xconsole
- tkmessage: Vertical replacement for xmessage
- tkprompt: Replacement for xprompt

<http://www.ubergeeks.com/~jmalone/tktools.tar.gz>