home

1. Description

My desktop, how I use it and how to set it up.

2. Usage (terminal)

I use *ksh(1)* with emacs–like keybindings. History is saved on `~/var/history/ksh.hist`.

My PS1 prompt has the following structure:

- The prompt sets the window title as the name of the tty device and the pwd (current working directory) separated by a colon and a space. The `/dev/` prefix is removed from the tty name; and the `$HOME` prefix is replaced with a `~` (tilde) on the pwd name. So if I'm on `/dev/ttyp2` at `/home/phil/tmp`, the title of the window will be `ttyp2 ~ ~/tmp`.
- The prompt begins with a newline, for displaying vertical space between the output of the previous command and the current prompt. This makes easier the prompt visual identification.
- If the previous command exits with non–zero exit status, this exit status is printed, as bold and red, at the beginning of the prompt.
- The left prompt consists of a single character. This character is white for normal users and red for the super user. The left prompt is padded at the left with a single space.
- The right prompt is the pwd with the `$HOME` prefix replaced with a `~`. The right prompt is padded at the right with a single space.

I use few aliases, most of them to force human–readable output. The most useful alias is `fuck`, which rerun the previous command with doas(1).

```
alias fuck='doas $(fc -ln -1)'
```

3. Usage (X11)

I use a blank desktop with no bars, docks or panels.

The first thing done when I log into an X session is to load the environment variables (by sourcing `~/.profile`) and to load the X resources with *expenv(1)* and *xrdb(1)* (expenv is needed to expand environment variables in the resources file). Key combinations are bound to commands by *sxhkd(1)*. The cursor is made invisible after a brief period by *unclutter(1)*. Music is managed by the *mpd(1)* music player daemon.

There are two ways to interact with the desktop: with the mouse (using a *pmenu(1)*), or with the keyboard (using *xprompt(1)*). Both ways output a command to a interpreter called *xinterp(1)*. Responses are output in a notification window managed by *xnotify(1)*. Each piece communicates with another by means of two named pipes, `$XNOTIFY_FIFO` and `$XINTERP_FIFO`.

3.1. Desktop Notification

The `$XNOTIFY_FIFO` environment variable contains the path to a named pipe read by a program called *xnotify(1)*. This program reads lines from its stdin and pops up a desktop notification for each read line. This named pipe is written by several scripts.

First, a script called notify is run in the background and checks the battery and computer temperature every minute. If the battery is below 20% or the temperature is above 60°C, a line is written into `$XNOTIFY_FIFO` and a notification is displayed for the user.

The *xinterp(1)* script also writes into `$XNOTIFY_FIFO` depending on the instruction it reads (see next section below). For example, when it receives an instruction to change the music, a notification containing the currently playing song is sent to `$XNOTIFY_FIFO`; when it receives an instruction to change the current desktop, a notification listing the desktops is sent to `$XNOTIFY_FIFO`.

3.2. The Desktop Interpreter

The `$XINTERP_FIFO` environment variable contains the path to a named pipe read by a script called xinterp(1). This script reads instructions from its stdin and run a given command depending on the instruction. Those instructions are generated by *pmenu(1)* via mouse interaction (see the Using the Mouse section below) or by *xprompt(1)* via keyboard interaction (see the Using the Keyboard section below).

For example, when selecting Firefox to be run using *pmenu(1)*, the command `run firefox` is sent to `$XINTERP_FIFO`. The `run` command opens a new instance of a program or focus an already existing one.

xinterp(1) understand several instructions, such as to launch an application, open a manual page with *zathura(1)*, change the current desktop, manipulate the current window, etc.

3.3. Using the mouse

When right clicking on the root window (ie', the desktop), a pie menu is displayed by *pmenu(1)*.

This *pmenu(1)* instance reads menu options from `~/var/cache/pmenu.cache`, and outputs instructions to `$XINTERP_FIFO` to be interpreted by *xinterp(1)*. The cache file is generated by *xinterp(1)* with the following command:

```
$ xinterp menu
```

With the pie menu, I can launch applications, change the current desktop, draw a terminal on the screen, change the song, take a screenshot, and control the active window.

3.4. Using the keyboard

After typing Alt+space, a prompt is displayed by *xprompt(1)*. This *xprompt(1)* instance reads options from `~/var/cache/xprompt.cache`, and outputs instructions to `$XINTERP_FIFO` to be interpreted by *xinterp(1)*. The cache file is generated by xinterp(1) with the following command:

```
$ xinterp prompt
```

With the prompt, the user can type any instruction that *xinterp(1)* understands, for example, open man pages.

3.5. The Window Manager

Windows are managed by the *shod(1)* window manager. It is an hybrid (tiling and floating) tabbed window manager that is controlled solely by responding to client messages with EWMH hints and regular ICCCM events, and by using a given key modifier with the mouse pointer.

Shod maintains one set of desktop for each monitor. Each set of desktop works like GNOME activities: The X session begins with a single desktop; when a window is created, this desktop is populated, and a new empty desktop is created. When an empty desktop is populated, a new empty one is created.

Windows begin floating on the desktop and are spawned in a proper empty place. The first window is spawned on the center of the monitor. The next windows are arranged in empty spaces on the monitor.

When a window is maximized, it is tiled. Tiled windows are organized in columns; each tiled window ocupies a row in a column. This columnated behavior imitates the way acme(1) handle its columns and frames.

4. Theme

Themes are installed mostly at the `~/theme` directory. I use the following themes.

- **Tango color scheme:** The 16 colors of my terminal emulator are those from the palette used by the Tango icon library. This palette is described in the wikipedia article *https://en.wikipedia.org/wiki/Tango_Desktop_Project#Palette*.
- **Retrosmart icon theme:** I use the Retrosmart icon theme, a set of icons mainly based on the Haiku OS look (*https://github.com/mdomlop/retrosmart–icon–theme*).
- **Retrosmart X11 cursor theme:** I use the white version with alpha shading of the Retrosmart X11 cursor theme (*https://github.com/mdomlop/retrosmart–x11–cursors*).
- **Input Mono Narrow font:** I use the Input Mono Narrow font from the Input family of fonts (*https://input.djr.com*).
- **Motif–like window decoration:** I use the default window decoration distributed with the shod window manager.
- **Black background:** I use a *xsetroot(1)* to set my background, with the options `–mod 3 3 –bg '#000000' –fg '#121212'`

5. Files

The contents of my `$HOME` directory are listed below. Note that I do not use the XDG home structure.

- `~/.profile`: Shell script that sets the environment variables necessary to set up a user session.
- `~/.session`: Shell script called by ksh(1) to set up a shell session (prompt, aliases, etc).
- `~/.xsession`: Shell script called by xenodm(1) to set up a X session (notification system, window manager, etc).
- `~/files/`: Where I archive documents, media, and files for consumption. Each category of files has a subdirectory in it. For example, `~/files/doc/` for non–fiction books; `~/files/lit/` for fiction books; `~/files/meme/` is the meme folder; `~/files/mus/` is the music folder; etc.
- `~/proj/`: Where I put whatever I am working on (mostly college stuff, things I am learning, and programs I write). Each project has a subdirectory in it. For example, `~/proj/xmenu/` for xmenu(1); `~/proj/c/` for notes on the C Programming Language; etc.
- `~/rules/`: Configuration files (aka dotfiles). For example, `~/rules/vimrc` (for vim); `~/rules/keybindings` (for sxhkd); etc.
- `~/skel/`: Skeleton files, that is, templates for different file formats. For example, `~/skel/Makefile` is a template for a Makefile; `~/skel/man.1` is a template for a section 1 man page; etc.
- `~/theme/`: Where I keep icons, cursors, fonts, etc.
- `~/tmp/`: Download directory and where I dump stuff. I try to keep it clean, but most of the time it is a mess.
- `~/usr/`: Program files (binaries, sources, manuals, etc). Each subdirectory contain files installed from a given source or using a given method. For example, `~/usr/local/` contains data for programs I install manually; `~/usr/python/` is for programs installed with pip(1); `~/usr/go/` is for programs installed with go(1); etc. Each subdirectory contains the directories `bin/`, `src/`, `man/`, and `etc/` (for binaries, source, manual and other files). For example, `~/usr/go/bin/lf` is the binary for lf installed by go(1); `~/usr/local/man/man1/xmenu.1` is the manual for xmenu(1) installed by its Makefile; etc.
- `~/var/`: Files that are not managed manually, but by applications. `~/var/trash/` is for trashed files managed by trash(1) and untrash(1); `~/var/mail/` is for email managed by mutt(1), mbsync(1) and others; `~/var/history/` is for command history from ksh(1), xprompt(1) and others; `~/var/cache/` is for cache used and generated by several programs.