Pining for the Fjords

> *Wichtig*
>
> This program, and most importantly, the data it generates, is **intended for your personal use only**. In jurisdictions recognizing rudiments of workers' rights, employers even asking employees to collect such data are breaking the law. In others, you should fight any attempt to impose the use of this or any other similar program.

Pftf is another take at time keeping and work time accounting. I wrote it when I finally got down to a 30h/week contract to see that I don't do too much overtime. While I was about it, I figured I could build this thing into something that would let me find out if all the organization and meeting business really is that much of a time sink.

Pftf lets you:

- Keep track of how much time you spent working on what,
- Figure out how much work you are/were supposed to do, taking holidays, vacation, sick time, etc., into account,
- Build graphs showing the acquired data.

Note that you will probably need to do some configuration to use pftf. See INSTALL and `pining --help-config`.

For the Impatient

After installation, simply run `pining` to create your .pftf directory.

Create a file ~/.pftf/config containing something like this:

```
[general]
wClasses = Vac, Sic, Tra, Qual, Dev, Data, Comm, Admin, Mail, Orga
hoursPerDay = 7.5
[gui]
width: 56
```

(you'll want to adapt this to your needs and likings). Run tkpftf, ideally telling your window manager to dock it. Click on the work classes as you go along doing your work.

To evaluate, run `pining` to find out the stats for the day, `pining week` or `pining month` for the respective time intervals. Add the `-p` option to get the distribution of time spent on the various wClasses over the time displayed.

At the beginning of the following month, run `pining -m -1 graph` to get a chart sowing the collected data in pined.pdf. Then run `pining archive` to get old data out of the way.

To find out about time tracking from the shell prompt, vacation, colour and, in general, what all this really means, read on.

Usage

Almost everything in pftf accesses the work time file (recording the work times, WTF in what's below).

There are three programs you will be using:

1. tkpftf -- a GUI for entering dates, times and labels to the WTF.
2. pineday -- a command line program for entering occupations filling a whole day into the WTF.
3. pining -- a command line program that evaluates the contents of the WTF.

tkpftf

This is the first interface to time tracking you should try. It is supposed to be dockapp-sized. The width of the window is controlled through the [gui] width setting, its height is then computed from the wClasses to be displayed and the [gui] labelFont setting.

You probably want to use your window manager to embed it in a dock, make it always-on-top, or similar. You can use the [gui] title setting to adapt the window title if necessary.

Btw., given the simplicity of the input format, it would not be hard to write alternative input UIs. Contributions are welcome.

pineday

Use this program to add entire days to your pining file. Here's the output of pineday --help:

```
usage: pineday [options] <label> <date> [<end-date>]
  adds day(s) to the work times.  Dates may leave out months and
  years for their current values and must otherwise be in ISO format,
  e.g., 15, 11-14, 2000-11-14

options:
  -h, --help  show this help message and exit
```

So, to say you'll be on vacation on the 15 of the current month, type:

```
pineday Vac 15
```

To say you traveled from October 20th through 25th:

```
pineday Trav 10-20 10-25
```

Pineday will not add holidays or weekends even if they are included in the range (if you really need to do that, you can only do it manually at this point).

The lines pineday leaves in the WTF are interpreted as meaning "Worked as many hours as specified in hoursPerDay on those days". If you log work on days added by pineday using the GUI, it counts on top of these hours.

pining

This is the main interface to the suite. Here's its built-in help:

```
usage: pining [options] [<action>] -- compute pining times.
 Actions include: week, quit, edit, graph, makecli, day, month, archive
    you can also pass a wClass as action.

options:
  -h, --help            show this help message and exit
  -a, --with-archive    add archived data for computation
  -d DAY, --day=DAY     output for DAY instead of current day
  -m MONTH, --month=MONTH
                        output for MONTH instead of current month
  -y YEAR, --year=YEAR  output for YEAR instead of current year
  -o FILE, --output-to=FILE
                        output graphics to FILE
  -p, --percents        output distribution by label
  -H, --help-config     Show help on items available for
                        /home/msdemlei/.pftf/config and exit
  -S, --help-holidays   Show holidays known to pftf and exit
  -C, --help-colours    Show available colours and exit
```

The actions mean:

- nothing given -- compute for the specified day
- week -- compute for the specified week
- month -- compute for the specified month
- graph -- create a graph for the specified month (requires pychart)
- edit -- bring up the timekeeping file in the editor specified in the environment variable VISUAL, locking it (see Locking)
- archive -- move all lines not pertaining to the current month to an archive file.
- a wClass or "quit" -- write a timestamp with the current label to the WTF, or end current label for quit.
- makecli -- see Alternative CLI

For most actions, by default the time spent and the time yet due will be printed. With -p, you get the distribution by label on top of that.

The day, month, and year options can be negative; they will then be interpreted relative, e.g.,:

```
pining -m -1 graph
```

will output the graph for last month,:

```
pining -d -1 -p
```

will show your work distribution for the last day. Note that the combination of negative days, months, and years may take some thought to understand and thus should be avoided.

`pining graph` will write a graph of the selected month's work times to a file; by default, this is pined.pdf. You can override this using the -o option and/or the defaultOutputName setting in the plot section; as is good custom, a dash means stdout.

`pining archive` will move all lines from some other month than the current one from the pining file to an archive location. You should to this monthly, since most operations pining does take the entire pining file into account. After data is archived, you must give the -a flag to include it in any computations. For example, after a `pining archive`, `pining -m -1 month` will not see any work done any more, you would have to say `pining -am -1`.

WTF format

Pftf doesn't use a database back end, which would be massive overkill for what is done here. Instead, there is a simple, flat text file.

There are times when you forgot to log out, break or unbreak or change a label in the GUI. When that happens, say `pining edit`. This brings up the editor you have configured in the VISUAL environment variable, and you can correct the timestamps. You should probably refrain from adding new lines altogether unless you are able to read and interpret error messages.

Note that if you set the [general] allowBlanksInLabels option to True (default is False), you must make sure your editor does not convert tabs to spaces (quite a few do unless you configure them).

There are three kinds of lines in the WTF:

- <timestamp><tab><label> -- these signify the start of a new time interval (and, if one was already open, the end of the last). The interval will receive <label> as label.
- <timestamp> -- signifies the end of an interval, either because a break started or because no label was selected.
- <date><tab><label> -- signifies a whole day was spent pining. Lines like this can be freely interspersed with the other two. This is used for vacation, travel, and similar things.

where <tab> may be a sequence of blanks if allowBlanksInLabels is False (the default).

Locking

To avoid messing up the pining file, it is locked while it is changed or read. Locking is done via symbolic links in the file system (you can see them in ~/.pftf). It may happen that a program malfunctions and fails to remove the lock. This is called a "stale lock". A program trying to lock the file will fail to do so, and you will have to remove the lock file manually.

However, make sure that the lock file really is stale. It's not always easy to do that, but under at least under Linux, the lock file really is a link pointing to the /proc entry of the locking process. You can use this to check what process holds the lock. Try, e.g.,:

```
pining edit
```

This will open your editor and leave the pinging file locked. In another window, you can try this again (the command will appear to hang for about 10 seconds):

```
msdemlei@victor:/home/msdemlei > pining edit
/usr/bin/pining: File locked.  Remove /home/msdemlei/.pftf/lock_pining.tsv if the lock is  stale
```

The interesting part is the path to the lock file given in the error message. If you're not sure why the lock is still there, you can do:

```
msdemlei@victor:/home/msdemlei > cat /home/msdemlei/.pftf/lock_pining.tsv/cmdline | tr '\000' ' '
/usr/bin/python /usr/bin/pining edit
```

(the tr magic makes the output a bit more readable) -- so, you see the link probably isn't stale but you forgot to close an editor. On the other hand, if the output is something like this:

```
cat: /home/msdemlei/.pftf/lock_pining.tsv/cmdline: No such file or directory
```

there's a good chance the lock is stale (of course, it could be held from a process on another machine; similarly, if some weird command line is given above, it's perfectly possible that the lock is stale and the process id of the crashed program has been reused; so, use your human intelligence).

Since it does lock handling, it's a good idea to use `pining edit` as opposed to editing ~/.pftf/pining.tsv directly to fix problems.

The timekeeping GUI will inform you if you try to change a locked file. As the error message states: The event will **not** be recorded.

Alternative CLI

If you want to set labels without using the GUI, you can use `pining <label>` or `pining quit` to stop. This may be convenient for automated logging, e.g., in logout scripts or when certain activities are linked to the execution of specific binaries. Also, you can bind such a command to keystrokes, probably in your window manager. The dockappish thingy picks up these changes by polling the WTF. The poll interval can be configured in [gui] pollInterval.

Since it's a bit of a waste to pull in the python runtime to produce a timestamp, you can make pining generate a binary tailored to your settings using `pining makecli`. This will leave an executable named `pinestamp` in your current directory, of about 4k on x86 architectures. Call it like pining, with either quit or a wClass.

If you change wClass or the name or location of the WTF, you'll have to rebuild pinestamp.

Note: pinestamp currently ignores locks and does not lock the WTF itself. I may change this at some point. Don't hold your breath, though.

To Do

Some things I'll want to see in pftf aren't yet there. Things that'll come include:

- less resource intensive "dockapp" -- a little C implementation, probably based on the window maker dockapp support code, would be nice. The python/tkinter stuff is too fat for a dockapp IMHO.
- Putting the date formats into settings was a folly. Undo.