SEPM/Linux Tasks Winter Term 2009

SE/Linux Team[*]

November 4, 2009

1 Task 2

1.1 Focus

This task focuses on designing an advanced client, designing good user interfaces and creating a better build system.

1.2 Mission

In the previous task you have gained important elementary know-how on the SEPM/Linux EHR system and you've become familiar with your new development environment and tools. You will now use this know-how to implement more components of the system: graphical or text-based clients for the physician, pharmacist and the patient.

1.3 Tasks

1.3.1 Designing and coding

Implement three separate client programs: one for the physician, one for the pharmacist and one for the patient. The required functionality of the different clients is described in [9]. As a rule of thumb, every call of the `Ehr::Provider` interface must be used in at least one of your clients.

- Decide whether you want to implement a graphical user interface or a console-based one.
  - Use GTKMM [7], Glade [6] and GladeMM [7] to implement a graphical user interface *or*
  - Use Curses [4] and its form, panel and menu libraries to write a console-based user interface

[*] `se-linux@inso.tuwien.ac.at`

  - Note: development with GTKMM is probably more productive thanks to the Glade RAD tool.
  - Also note: regardless of which approach you use a command-line interface like in the previous task is *not* required anymore.

- Implement some kind of *settings* dialog that is used to configure values that usually don't change while the software is running (does not apply for example to the path to the patients certificate in the pharmacist's client software).
- Implement persistent storage of settings. A good option is to store them in a hidden file in the users home directory. With GTKMM you are encouraged to use the gconf [5] configuration system and it's C++ bindings GConfMM [7].
- Implement some kind of form to compose a document.
- Integrate your CairoMM rendering class to implement an *export document to PDF* feature.
- With GTKMM, use your CairoMM rendering class to implement a *view document* feature.
- It is important for the user interfaces of your clients to be usable. Let yourself be inspired by state-of-the-art graphical user interfaces. A good example for a console based user interface is the e-mail client Alpine [3], the successor to Pine [10].
- Although you have to deliver three separate programs, you are not prevented from sharing code between them. To facilitate reuse:
  - As usually, use classes.
  - Implement complex dialogs and complex widgets as classes.
- Stuff that is hard to figure out for the user has to be documented in a `README` file.
- Bonus task (bonus points): in the physicians client software, use libxml++ [8] to extract the patients personal data from his PEM file.

1.3.2 Building your programs

You already know hand-written Makefiles from the previous task. In this exercise you will use the (in)famous *GNU build system* (the *Autotools*) to reduce the complexity of maintaining Makefiles.

- Read the GNU Autoconf documentation at [1]
- Read the GNU Automake documentation at [2]
- Create an advanced build system for your sources

- Manually create a Makefile for building your client. Manually means that no tools may be used for creation of the file. Call your Makefile `Makefile.man`. It would be best to take the Makefile from Task 1 and extend it. Don't forget to keep an eye on the dependencies of the targets.
- Set up a GNU Autotools build system:
  - Use Automake.
  - Use Autoconf.
  - You don't need to write M4 macros. Find the ones you need on the internet.
  - The exercise tarball contains an M4 macro for detecting Ice.
  - Your build system will be tested using the target `distcheck` so make sure `make distcheck` works on all machines your code has to compile on. *If this check doesn't work you are not done yet!*
- Your build systems (manual and Autotools-based) need to work at least on the Linux and BSD shell servers.
- Versions of libraries might differ on the target platforms. They might even be incompatible or not installed at all. Use conditional compilation as work around if needed.
- `-Wall` is your friend, keep using it.

1.4 Deliverables

Tag the source code in your subversion repository as `/submission/task2_final`.

1.5 Hints and pointers

Remotely using graphical programs To be able to run graphical programs on a remote computer you need to do enable what is called *X11 forwarding.* This means that a remotely started graphical program will be displayed on the local display.

To log into a remote machine enabling X11 forwarding type:

```
ssh -XCY l1234567@lbzux.inso.tuwien.ac.at
```

You can now start graphical programs on host lbzux while they are displayed on your local X server.

References

[1] Free Software Foundation: *GNU Autoconf – Creating Automatic Configuration Scripts.* `http://www.gnu.org/software/autoconf/manual/`. Version: 2007. – [Online; accessed 2007-02-26]

[2] Free Software Foundation: *GNU Automake.* `http://www.gnu.org/software/automake/manual/`. Version: 2007. – [Online; accessed 2007-02-26]

[3] Office of UW Technology: *Alpine Messaging System.* `http://www.washington.edu/alpine/`. Version: 2009. – [Online; accessed 2009-04-06]

[4] Raymond, Eric S. ; Ben-Halim, Zeyd M.: *Writing Programs with NCURSES.* `http://web.cs.mun.ca/~rod/ncurses/ncurses.html`. Version: 2007. – [Online; accessed 2007-02-26]

[5] The Gnome Project: *GConf configuration system.* `http://www.gnome.org/projects/gconf/`. Version: 2008. – [Online; accessed 2008-04-20]

[6] The Gnome Project: *Glade - a User Interface Designer for GTK+ and GNOME.* `http://glade.gnome.org/`. Version: 2008. – [Online; accessed 2008-04-20]

[7] The gtkmm Team: *gtkmm – the C++ interface to GTK+.* `http://www.gtkmm.org`. Version: 2007. – [Online; accessed 2007-02-26]

[8] The libxml++ Development Team: *libxml++.* `http://libxmlplusplus.sourceforge.net/`. Version: 2007. – [Online; accessed 2007-02-26]

[9] The SE/Linux Team: *The SEPM/Linux EHR System.* `http://se-linux.inso.tuwien.ac.at/se1/tasks.phtml`. Version: 2008. – [Online; accessed 2008-04-20]

[10] UW Technology at the University of Washington: *Pine – a Program for Internet News & Mail.* `http://www.washington.edu/pine/`. Version: 2009. – [Online; accessed 2009-04-06]