Teaching and **examining** with Netbeans IDE

Pieter van den Hombergh

Fontys Hogeschool voor Techniek en Logistiek
Venlo

February 10, 2015

Content

No one can write properly these days. . .

- Proper and beautiful writing with a quill was once an art with much appreciation.
- Nowadays you can even teach without having a proper handwriting.
  - Living proof is in front of you. . .
- So who am I to expect my students to have a proper hand in writing?
- Note: using a pencil and paper [1] to make a sketch before you start writing/coding, even if you throw the sketch away in the process is still a best practice.

[1] White board, whatever

In practice you use an IDE

- I have seen the days of punch cards and paper tape.
- Programming was programming in the small. (As was examination).
- Nowadays you let students use an IDE during their practical work
  - For didactic purposes: You can play with the things you make. e.g. For Java: bluej or greenfoot.
  - To enhance productivity, get bigger jobs done: **Netbeans IDE**.
  - Advantage of Netbeans: pure java, runs on everything (even **Raspberry Pi**)
- Nowadays programming is programming in the large, know and understand the important APIs and how to use them. Read (java)doc etc.
- Logical consequence: Use and IDE (any of the above) during exams too.

Authenticity, Logistics, tooling

- Examination requires certain conditions.
  - Students are not allowed to share work (cheating), where in lab hours they are advised to do so.
  - Only certain resources or references are allowed, where as in lab hours **google** and stack overflow are available.
  - **Solution** computers that are controlled by the examiner and are isolated in a examination network.
- Logistics: How to collect the exam work?
  - Students must be able hand in, but need to stay isolated from one another.
  - Collecting the work from the workstations?
  - Use a upload facility: student private repository hosted on a server in the examination network.

From CVS to USB

And all in between.

2005 The beginning.

- Labs with a computer per student during exam sessions.
- Multiboot workstations in lab. Separate exam partition/os. Restricted shell (rbash). Linux (SuSE) based.
- Repository: **CVS** server on one workstation in the same lab.
- Students where well versed (well, almost) in CVS.

2009 Moved subversion.

2012 Bring your own device.

- Mix of lab PCs and laptops. How to control the laptop during exam?
- Boot from USB stick.
- Still using a repository.

2015 Dropped repository. Collect exam from sticks.

Lessons learned

- An exam is not exam without proper surveillance or logging; it is a lab session at best. Consider:
  - How to prevent students to collaborate?
  - Or share exam credentials?
  - Or booting something else then the proper exam setup?
  - Accessing local disk, network at non allowed places?
- Personally I never value or grade work handed without some proof of authenticity. Of course feedback still applies.

Moore's law to the rescue

- Dropping prices of USB storage make things affordable.
  - Required storage space on USB stick: less then 4GB, including storage for user workspace/home dir.
  - €3.34 (2015-02-10) for 4GB.
- Speed up, USB 3.0 is a big productivity boost.
  - Lowest capacity usable and available 16GB.
  - Price is about 5 fold. Speed is 10 fold.
  - Good experience with **Sandisk 16GB Extreme USB 3.0.**

- Use of fitting USB hubs also works.
- 7 port ANKER USB 3.0 hub gives good results.

Parallel programming, differently

Other resources: 100 USB sticks

It still amounts to about €1600,– of flash drive.

USB recipe, ingredients

Distribution Ubuntu based host OS.

- Up to date kernels, more chance to use with new laptops.
- Otherwise quite stable
- Start with `ubuntu-mini-remix-14.04.1-amd64.iso` from `http://www.ubuntu-mini-remix.org/`
- Use lightweight desktop like XFCE4.
- Current version 14.04.1 LTS.

Apps Mix and match whatever you need. Most of the time with a simple `apt-get install` sometimes special PPAs (e.g. Java 8).

Home grown packages Maybe some deb [re]packaging required: Netbeans, glassfish etc.

Packaging is simple: Pickup an installed version of the app, add some meta info and checksum the lot. See script sample.

USB recipe, Tools

Host computer running same OS version. This eases work. Tweak the max number of boot devices (add a parameter to grub command line).

Customizer Simple but adequate. Version 3.2.3.

- Needs some tweaks to get it running. (Gambas 3 vs 2 on U 14.04).
- `https://github.com/clearkimura/Customizer`.
- Revived. They seem to be moving to python.

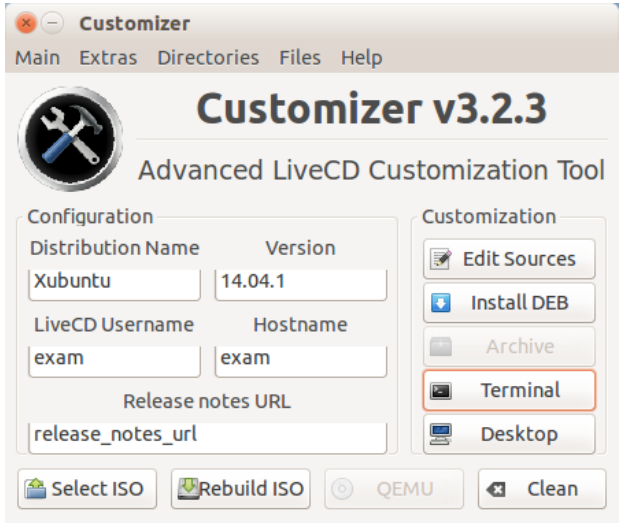
Startup Disk Creator (`/usr/bin/usb-creator-gtk`), standard in Ubuntu.

DD Good old `dd`.

Cook A good Linux Cook  . Some knowledge is indispensable.

Preparing the dough

- Prepare basic stick.
  - Load iso file with customizer. This will unpack iso into `/home/FileSystem` and live system helpers into `/home/ISO`
  - Open **terminal** in customizer. This does 'apt-get update' automatically.



- In the terminal you have all apt-tools. You are in a `chrooted` environment
- Install (apt-get) whatever you want, e.g. desktop and window manager (e.g. XFCE4) and any apps you like.
- Install local `.deb` packages with 'install deb package'.
- Rebuild ISO which creates a new iso file in `/home/`.
- Rename the iso, e.g. add a date to it. This ISO is also a snapshot, making it easy to revert your steps.

Make a sticky

- Add exam user account and settings (e.g. 'empty password').
- Add correctors account if required.
- Add everything you need in the next steps to the `/home/FileSystem`. That will land in the ISO and stick too.
  Example: JDK documentation for on stick browsing during exam.
- Bake one stick first. Simply use **Startup Disk Creator** from the Host menu. Allow for some work space. 1GB (the default) is typically just fine.
- Test it.
- If satisfying, continue to the next step, otherwise do some more tweaking in customizer as required.
- You could provide the ISO to the students, so they can practice with the USB and IDE on their device. I offered to create (batches) of sticks, owned by the students.

Make it exam proof

- Remove add kernel modules and drivers not needed.
  - Network[2], wifi, bluetooth, local hard drives.
- Remember to do a `depmod` -a inside the **customizer-terminal**. Otherwise your stick will not boot.
- Make a new ISO. Rename it properly.
- Test again of course.

[2] You can leave wired network in, if you want a network based exam

Prepare exam workspace

- Bake a new stick and boot from it on exam machine to *prime* it. This will create the exam user space under `/exam/exam`.
- Tweak settings and such (UI menu to make it less confusing, remove unneeded desktop icons, make browser bookmarks to local documentation, e.g. to JDK doc.
- Shut down exam test machine.
- Mount the primed stick with

```
cd /media/yo
mkdir casper-EXAM101
mount -o loop EXAM101/casper-rw casper-EXAM101
```

- Save the primed data **exam work space** from `exam/exam` to some place safe for later use (stick cleanup). Use e.g. `tar czf` or zip. Result: `skell.tgz`

Stir, bake at 5 Volts, 14 per batch

- Make a dd-copy of the primed stick and save it under a **proper-name**.img

```
# note that count will depend on actual use.
# use dd to find the used space on the exam stick
# and round up.
dd if=/dev/sdc of=exam-20150210a.img bs=1M count=2600
```

- (sticky) Label your sticks if you not already did so.
- Now connect your USB hubs and
- Insert your sticks, ensuring the host sees it. (as in: wait a bit between sticks).
- Insert in label order. In this phase it ensures proper matching of stick label and stick disk-label (EXAM101).
- start the mkExamSticks script with appropriate parameters

```
# create sticks from exam image
sudo ./mkExamSticks -d c -l 100 -c 14 -i exam20150210a.img
```

This writes and labels all 14 sticks (the c parameter) in *parallel*, which takes about 90 seconds on my machine.

Core of bake script

The speed-up lies in the way you start `dd` sub processes and the extra parameters to dd.

Listing 1 : Tiny details count here

```
# first ensure all sticks are umounted
for i in $(seq 0 $((${COUNT}-1))) ; do
    disk=/dev/sd$(add2ascii ${DISKBASE} ${i})
    echo umounting ${disk}*
    ${debug} umount ${disk}*
done

for i in $(seq 0 $((${COUNT}-1))) ; do
    disk=/dev/sd$(add2ascii ${DISKBASE} ${i})
    printf -v LABEL "${NAME}%3.3d" $((${EXAMBASE}+${i}))
    ${debug} dd conv=fsync if=${IMAGE} bs=1M of=${disk} && \
    ${debug} mlabel -i ${disk}1 ::${LABEL} &

done
wait # join for java programmers
for i in $(seq 0 $((${COUNT}-1))); do
   disk=/dev/sd$(add2ascii ${DISKBASE} ${i})
    ${debug} umount ${disk}*
done
```

The topping, the exam tasks

- In our setup, the students receive a stick, personalized, with the appropriate netbeans project pre installed in `/exam/exam/Desktop/examproject` or similar.
- We prepare the exam from a solution project
  - adding corrector tags and TODO tags (manual labor). Here the corrector's workbench Netbeans plugin might be handy.
  - strip the solution
  - Make personalized copies with one copy per stick (e.g. EXAM101), this is the **local sandbox**.
  - Put them in a repository, one per student, if you so require.
- Insert the sticks in the hubs, connected, the disk label now identifies the sticks.
- Use the `primeSticks` command, no parameters; it can find how many sticks are mounted.
  - Installs fresh copy of the exam work space.
  - rsyncs local sandbox into exam work space.

The proof is in the eating

- Take the exam.
- Collect the sticks.
- *Harvest* the student projects.
  - Insert the sticks into the hubs
  - use the command `harvestSticks` to rsync the exam stuff back into the local sandbox.
    - Takes about 20 seconds for 14 sticks.
    - Inserting and removing sticks takes about the same time.
    - With 2 pairs of hubs your could speed up even a bit more.
  - Commit the work into the repository.
- Make the exam work available to the correctors.
  - Not netbeans plugin here yet  .
  - Now using simple PHP based web app, to allow parallel corrections by multiple correctors

Current use and recent developments

- It look as if the development of **Customizer** has been taken up again. Must have a look into it.
- In Venlo we do not use the repository anymore during the exam.
  - The students are relieved of sometimes error prone checking out and committing.
  - Less technical problems with modern laptops and Ethernet drivers.
  - Some laptops (Think Bitten Fruit Shop in Cupertino, CA) do not have an Ethernet connector anymore.
  - Less possibility of cheating  .
  - Although handling all sticks two times per exam, the overall logistics is simpler and quicker

Improvements and wishes left open

Simplification Prime the sticks generically, with just student ID, not student meta data.

- + Simpler handing out of sticks.
- + Results in less unused sticks.
- - make sure you get the association between stick and student right. E.g. let them add their id (student number) somewhere or register the issued sticks when handing out.

Security and authenticity Somehow protect the stick from tampering with e.g. a Linux box during an exam. Anything I can do, a smart student could do to. 

- + makes cheating more difficult.
- + Simple solution: Make appearance of desktop clearly recognizable and unpredictable
- - Would need some kind of encryption. Where to store they key?
- - probably would impair priming of sticks
- - no solution here yet; not yet seriously investigated.

Closing remarks

- You cannot organize a real exam without supervision. There will always be students that try to cheat when given the change.
- All presented here is freely available, either by license (E.g. netbeans) of by being OSS. See github for some versions of the scripts.
- In general, the students consider this kind of exam setup *fair* and **more realistic** then a paper programming exam.
- The setup is not restricted to Java programming. We do database exams in the same way.
- (Libre)Office is also installed, so if student handwriting is the problem in written exams, consider using an office document or `.txt` file as exam document.

References

Customizer	https://github.com/clearkimura/Customizer
Mini remix	http://www.ubuntu-mini-remix.org/
Github	https://github.com/homberghp/correctorsworkbench
Me	email:p.vandenhombergh@fontys.nl