

Flow, State and Persistence



*...or how I learnt to love change
and forget about it*

Outline

- ~ State and persistence: Files and Images
- ~ Different kind of states: Saving History
- ~ Distributing State ?

Flow ?..

- ~ Interruption is bad: it breaks the “flow”
- ~ ...yet (most) of our computers are not designed to prevent it
- ~ e.g. warnings that can be too disruptive
- ~ but one of the biggest problems is that state is not saved properly

State

- ~ **We do not like change: I want to find things as I left them**
- ~ In an usual work setup, everything changes: the state of your program, the position of your windows, etc
- ~ It's dramatically worse when you are working on more than one computer

Ideally...

- ~ Everything would be saved precisely as it is when leaving, and could be restarted to reach the exact same state
- ~ And I mean,.. everything. My debugger's state as well as my windows position ! I want true persistence.
- ~ Ideally ? well, it exists in Smalltalk...

Smalltalk

- ~ Smalltalk is an Object-Oriented language and environment.
- ~ Smalltalk does not deal with source files, but with an “Image”: a complete snapshot of a running system
- ~ Not so difficult in Smalltalk as everything is an object, you “just” need to implement persistence for the objects.

Consequences of Image

- ~ When you quit a Smalltalk session and you start it again, you will find things exactly as they were -- even if you were in the debugger...
- ~ You can have “projects” that are simply a bunch of objects serialized. I.e., you can come back to a project you worked with months ago and find everything as you left it.

Crude Persistence

- ~ The “right” model (for true persistence) would be the image model. But most applications/environments instead use a very crude way of saving state: **files**.
- ~ In fact, you save only the content of your work, not the complete state of your work environment

Current “Solutions”

- ~ What can you do to approach the ideal ?
 - ~ Keep your computer on all the time
 - ~ Use the “sleep” mode
 - ~ Use sessions / virtual desktops (unix)
 - ~ Use VMware and similar virtualization solutions
 - ~ Backup with things like Ghost
- ~ Basically, you simulate an image.

Distributed State?

- ~ Things are even more dramatic if you need to keep state among different computers.
- ~ Basically, no solutions really exist to do that — not surprising considering no good solutions exist for “freezing” state in the first place

But

- ~ If all your “state” is saved in files, you can synchronize multiple copies easily with programs like rsync
- ~ Just run rsync periodically (crontab) or on-demand to synchronize your work
- ~ You could also use a distributed file system such as Coda. Though, rsync is probably easier to setup.

rsync

- ~ rsync does incremental synchronization
- ~ `rsync -avz --delete-after
myDirectory/ remoteHost:~/
myDirectory-backup`
- ~ see <http://samba.anu.edu.au/rsync>

Another kind of state

- ~ As we saw, the capacity of “freezing” a current state is generally not well done (euphemism) in so called “modern” operating systems, and not very flexible
- ~ Indeed the only thing you do is saving the content of your work, but not the whole environment. Is it done properly at least ?
- ~ Well, no. There is some information that we lose all the time without thinking about it much: history.

History

- ~ I want to save my work at each step :
 - ~ I can use numerous copies of my files
 - ~ not very flexible
 - ~ difficulty of managing all the files
 - ~ difficulty of exploiting them
- ~ I want to possibly have different “branches” of my work (e.g. to try ideas)

Solution: Versioning

- ~ A versioning system such as Subversion will let me:
 - ~ save “snapshots” of my work
 - ~ let me access the complete history
 - ~ let me create “branches”
 - ~ I will also be able to keep synchronized different computers !

Versioning Systems

- ~ They are also ideal for working with other people
- ~ merge capacities, branching..
- ~ They should be mandatory :-) and integrated in the OS / Applications

Subversion examples

- ~ `svn add myFile`
- ~ `svn commit myFile -m "first version"`
- ~ `svn log`
- ~ `svn update`
- ~ `etc.`
- ~ see <http://svnbook.red-bean.com> for more documentation

Another example: restoring state when programming

- ~ When programming, you often need to debug or test a specific feature / behaviour
- ~ Create a proper testcase:
 - ~ basically you will “restore” a state...
 - ~ the more time you need to test things, the more writing such a testcase program will be worthwhile.

Conclusion

- ~ State is not well preserved on current Operating Systems, and not in a very flexible way (files...)
- ~ Things need to be better. We can take inspiration from Smalltalk.. sigh. After all it already did it properly 26 years ago....
- ~ Though, tools like subversion or rsync help a lot when you need to “find things as they were” and deal with history.