Version Control, Issue Tracking, Build Management

Software Engineering and Scientific Computing

Exercises First Day

Hanna Remmel
Institute of Computer Science
Im Neuenheimer Feld 326
69120 Heidelberg, Germany
http://se.ifi.uni-heidelberg.de
valtokari@informatik.uni-heidelberg.de

Contents

- Version Control
  - Tool Subversion SVN

- Issue Tracking
  - Tool BugZilla

- Build Management
  - Tool CMake

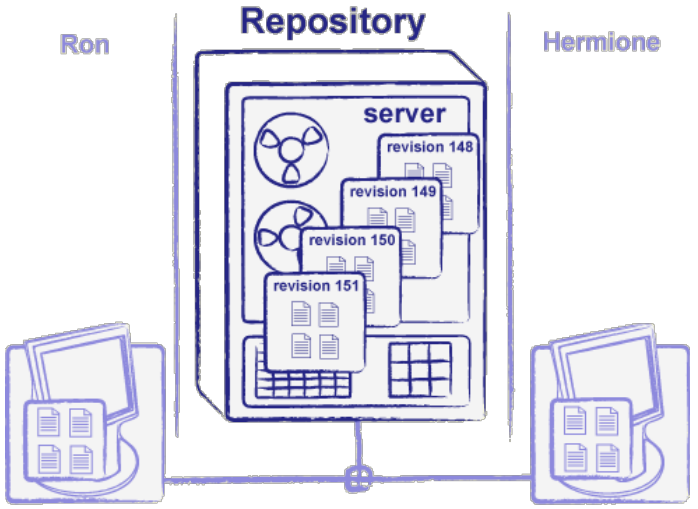
Question

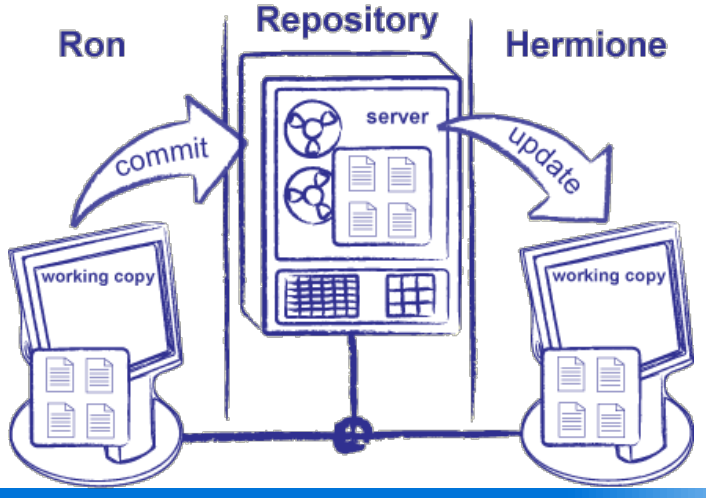
Do you use a version control tool?

If yes, which?

Basic Terminology

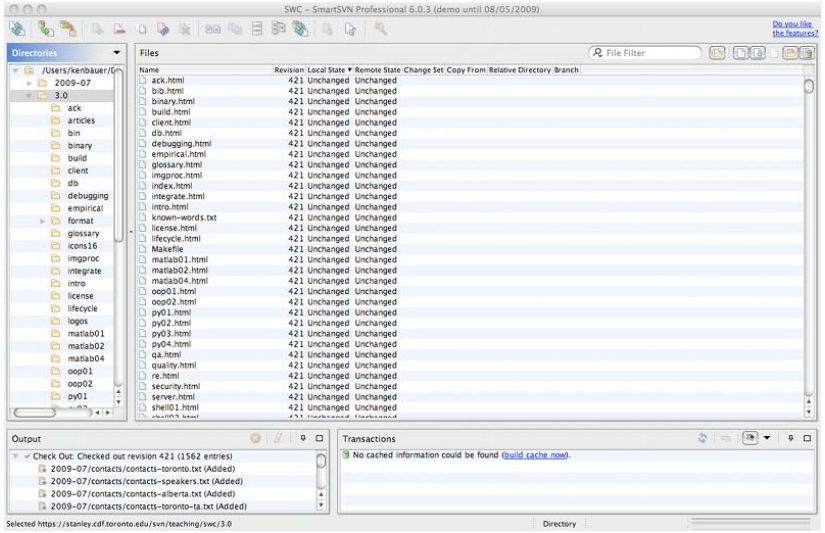
- Repository
- Revisions
- Working copy
- Actions
  - `checkout`
  - `update`
  - `commit`

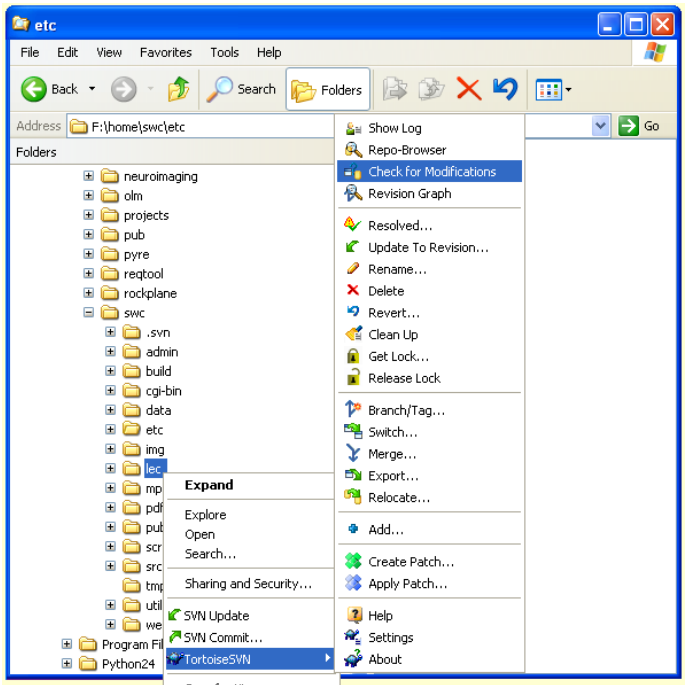




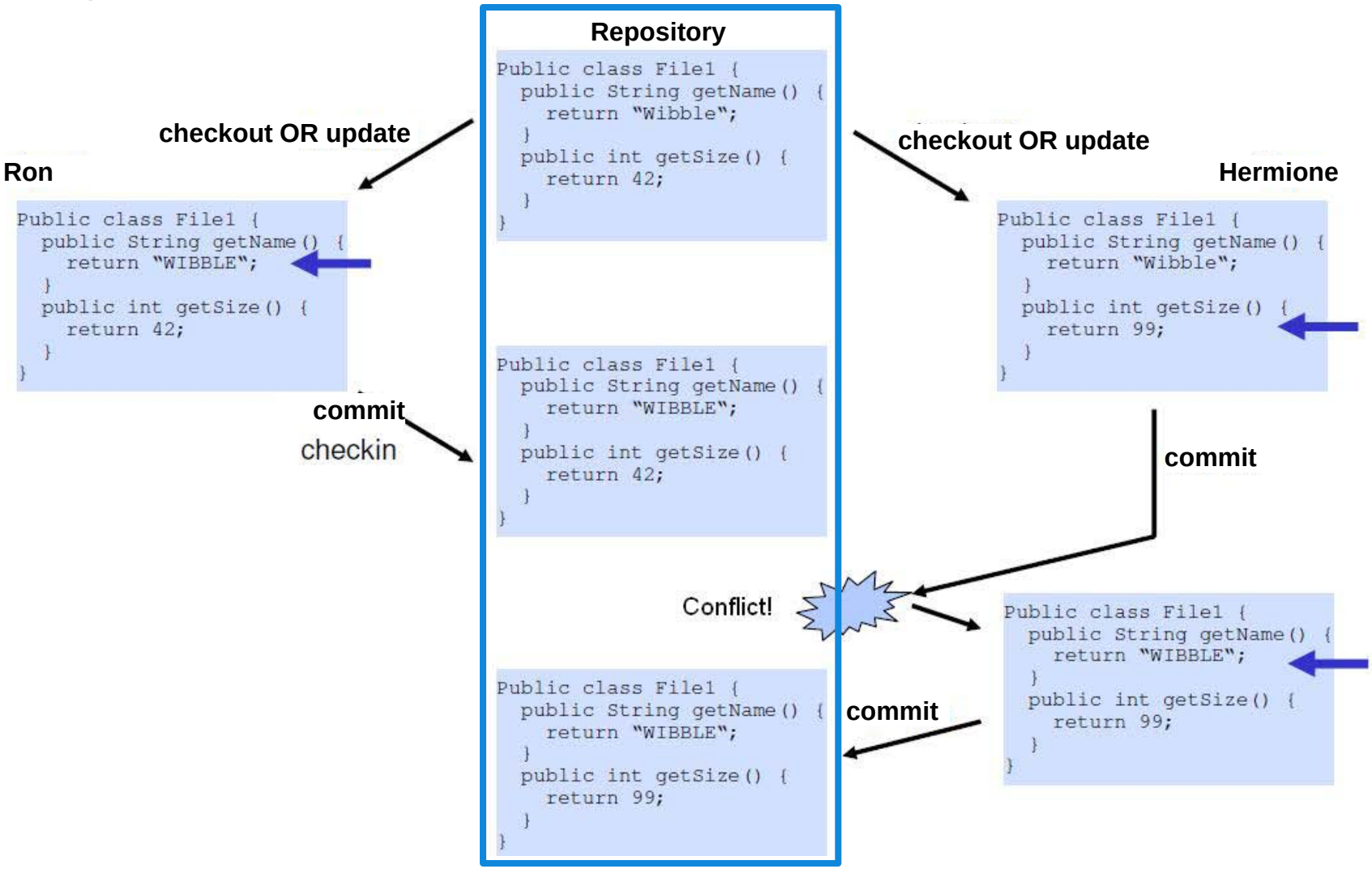
How to Do It

- One way to use Subversion is to type commands in a shell
  - A lowest common denominator that will work almost everywhere
- [SmartSVN] is a GUI that runs on Windows, Linux, and Mac (and anything that runs Java 1.4). It also provides Explorer/Finder integration
- [TortoiseSVN] is a Windows shell extension
  - Integrates with the file browser, rather than running separately





Example with a Conflict



Example of Resolving a Conflict

- Subversion puts Hermione's changes and Ron's in `file1.cpp` (Hermione's local copy)
  - Adds conflict markers to show where they overlapped

    `<<<<<<<` shows the start of the section from the first file

    `=======` divides sections

    `>>>>>>>` shows the end of the section from the second file

- Subversion also creates:
  - `file1.cpp.mine`: contains Hermione's changes
  - `file1.cpp.151`: the file before either set of changes
  - `file1.cpp.152`: the most recent version of the file in the repository
- At this point, Hermione can:
  - Run `svn revert file1.cpp` to throw away her changes
  - Copy one of the three temporary files on top of `file1.cpp`
  - Edit `file1.cpp` to remove the conflict markers
- Once she's done, she runs:
  - `svn resolved file1.cpp` to let Subversion know she's done
  - `svn commit` to commit her changes (creating version 153 of the repository)

Subversion Command Reference

Name	Purpose
`svn add`	Add files and/or directories to version control.
`svn checkout`	Get a fresh working copy of a repository.
`svn commit`	Send changes from working copy to repository (inverse of update).
`svn delete`	Delete files and/or directories from version control.
`svn diff`	Shows changes for directories/files in a unified diff format.
`svn help`	Get help (in general, or for a particular command).
`svn log`	Show history of recent changes.
`svn merge`	Merge two different versions of a file into one.
`svn mkdir`	Create a new directory and put it under version control.
`svn rename`	Rename a file or directory, keeping track of history.
`svn revert`	Undo changes to working copy (i.e., resynchronize with repository).
`svn status`	Show the status of files and directories in the working copy.
`svn update`	Bring changes from repository into working copy (inverse of commit).

Question

Do you use an issue tracking tool?

If yes, which?

Why Issue Tracking?

- One system to collect information about bugs and enchancements
- Organize your work in a development team
  - Visibility for the whole team
    - Who is working on which issue?
  - Prioritize issues
    - What should i do next?
  - Dependencies between issues
    - How much work is still done for the next release?
  - History of issues kept in a searchable location
    - Didn‘t we have this problem already in the past?
    - Why is this feature implemented like this?
- Communication
  - All information to one issue collected on one place
  - Information is accessable for all, better than email

Bug vs Enhancement

- Issue = bug or enhancement
- Differences between a bug and an echancement
  - Responsibility: Enchancements are payd by customer, bugs are payd by development
  - Documentation: specification needed?
  - Decision making
  - Priority
- Not every bug is a bug, it might also be a
  - User mistake: e.g. correct driver not installed
  - Faulty operation: user did not follow the advices in handbook
  - Missunderstanding: user expects a behavior that was not implemented
- Issue based Development
  - Developers work is based on an issue that is assigned to her/him
  - No issue – No working!

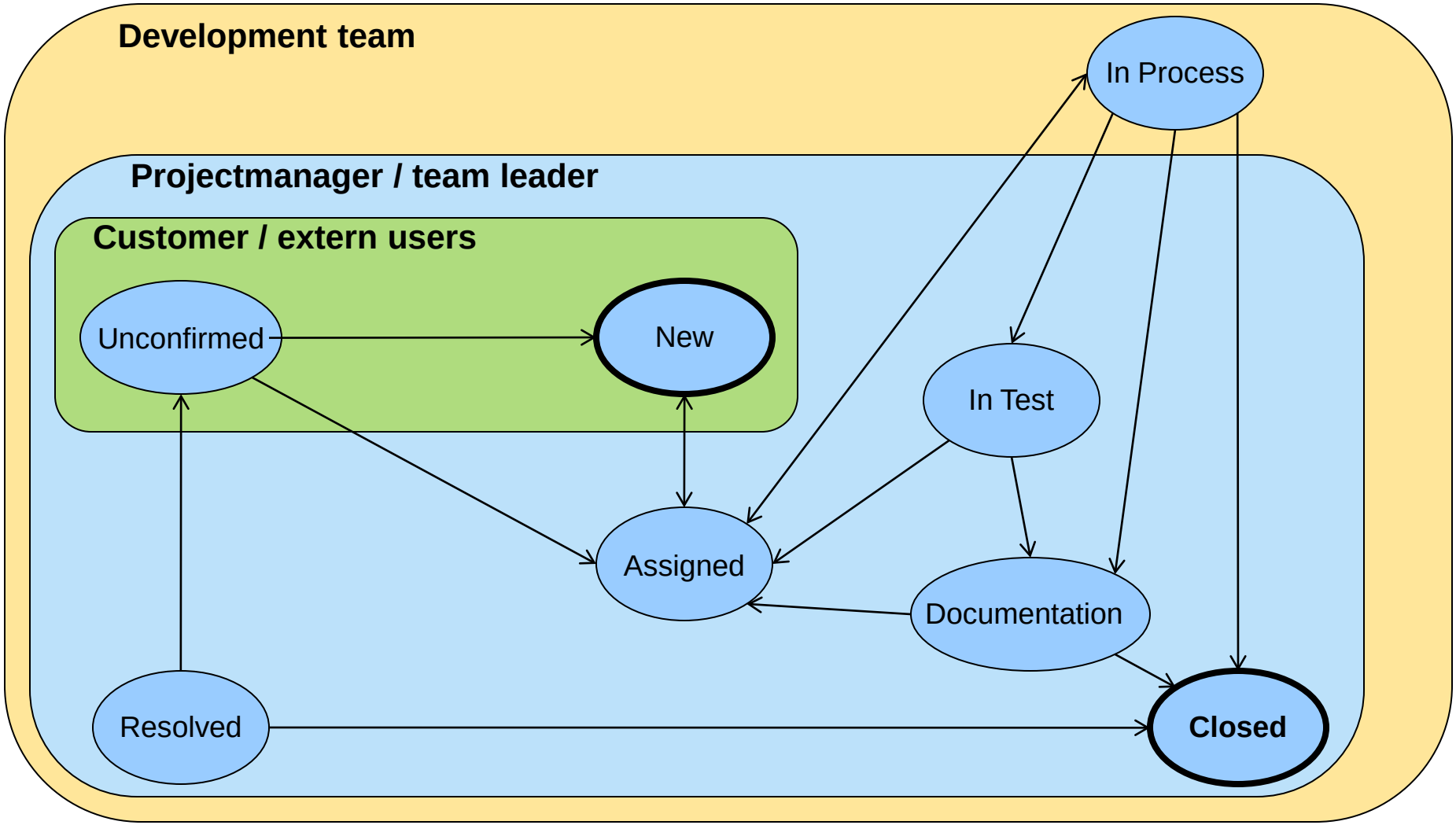
Attributes

- Issue Attributes include detailed Information about
  - Product, Component, Version,…
  - Priority
  - Target Milestone
  - Status
  - Assigned to
  - Summary (short description)
  - Description
- Description of an issue should be precise and include enough information
  - Bug: which steps has to be taken to reproduce this bug?
  - Enchancement: what is needed and how exactly should the new feature work?

Development Process: Status and Roles

- Issue goes through a defined development process
- Status of an issue specifies
  - Which steps has already been done
  - Which steps are to be done for this issue right now
- Status-Chances may only be allowed to specific roles
  - Project manager
  - Development team
  - Customer

Example: Status and Roles



Question

Do you use an build management tool?

If yes, which?

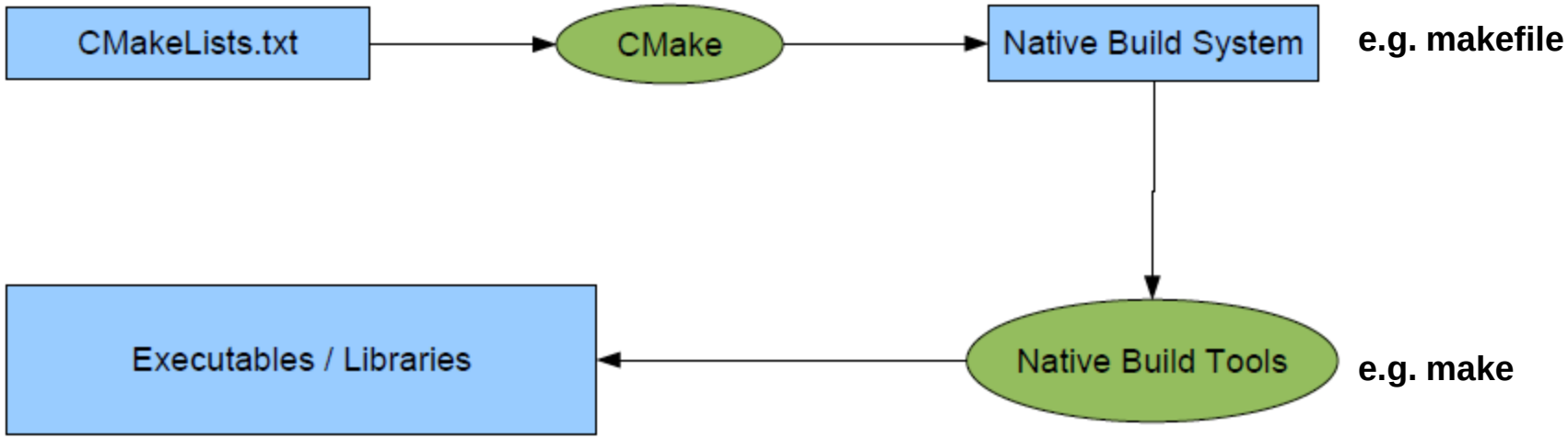
“software build refers either to the process of converting source code files into standalone software artifact(s) that can be run on a computer, or the result of doing so” (Wikipedia)

Build Management

- Every repetitive task is done through the build system
- Make is most widely used build tool that has
  - A way to describe what things to do
  - A way to specify the dependencies between them
- makefile
  - is needed when you compile your source code
  - The syntax is not easy to understand
- CMake: a tool to easy makefile creation
  - Cross-platform

CMake process

- CMakeLists.txt in every source file folder contains the project parameters and dependencies
- `cmake` creates a makefile that can be used for the native build tool



Example

MakeFile

```
INPUT_DIR = /lab/gamma2100
OUTPUT_DIR = /tmp
CHEMICALS = hydroxyl methyl
SUMMARIES = $(addprefix ${OUTPUT_DIR}/,$(addsuffix _all.csv,${CHEMICALS}))
all : ${SUMMARIES}
${OUTPUT_DIR}/%_all.csv : ${OUTPUT_DIR}/%_422.csv ${OUTPUT_DIR}/%_480.csv
	@summarize $^ > $@
${OUTPUT_DIR}/%.csv : ${INPUT_DIR}/%.dat
	@dat2csv $< > $@
clean :
	@rm -f *.csv
```

CMakeLists.txt

```
include_directories(${CMAKEDEMO_SOURCE_DIR}/w01-intro)
link_directories(${CMAKEDEMO_BINARY_DIR}/w01-intro)

add_executable(cdemo cdemo.c)
target_link_libraries(cdemo m)
set(PROGRAMS oglfirst pointers)
set(CORELIBS ${GLUT_LIBRARY} ${OPENGL_LIBRARY} m)
foreach(program ${PROGRAMS})
	add_executable(${program} ${program}.cpp) t
	arget_link_libraries(${program} ${CORELIBS})
endforeach(program)

add_library(geometry geometry.cpp)
add_executable(test_geometry test_geometry.cpp)

target_link_libraries(test_geometry ${CORELIBS} geometry)
```

Unfortunately i do not have an example of the same system in both syntaxes… :(

CMake Syntax

- # This is a comment
- Commands syntax: COMMAND(arg1 arg2 ...)
- Lists A;B;C # semi-colon separated values
- Variables ${VAR}
- Conditional constructs
  - IF() ... ELSE()/ELSEIF() ... ENDIF()
  - Very useful: IF(APPLE); IF(UNIX); IF(WIN32)
  - WHILE() ... ENDWHILE()
  - FOREACH() ... ENDFOREACH()
- Regular expressions (check CMake FAQ for details...)

CMake Commands

- INCLUDE_DIRECTORIES(“dir1” “dir2” ...)
- AUX_SOURCE_DIRECTORY(“source”)
- ADD_EXECUTABLE
- ADD_LIBRARY
- ADD_CUSTOM_TARGET
- ADD_DEPENDENCIES(target1 t2 t3) target1 depends on t2 and t3
- ADD_DEFINITIONS(“-Wall -ansi -pedantic”)
- TARGET_LINK_LIBRARIES(target-name lib1 lib2 ...) Individual settings for each target
- LINK_LIBRARIES(lib1 lib2 ...) All targets link with the same set of libs
- SET_TARGET_PROPERTIES(...) lots of properties... OUTPUT_NAME, VERSION, ...
- MESSAGE(STATUS|FATAL_ERROR “message”)
- INSTALL(FILES “f1” “f2” “f3” DESTINATION .)
- DESTINATION relative to ${CMAKE_INSTALL_PREFIX}
- SET(VAR value [CACHE TYPE DOCSTRING [FORCE]])
- LIST(APPEND|INSERT|LENGTH|GET|REMOVE_ITEM|REMOVE_AT|SORT ...)
- STRING(TOUPPER|TOLOWER|LENGTH|SUBSTRING|REPLACE|REGEX ...)
- SEPARATE_ARGUMENTS(VAR) convert space separated string to list
- FILE(WRITE|READ|APPEND|GLOB|GLOB_RECURSE|REMOVE|MAKE_DIRECTORY ...)
- FIND_FILE
- FIND_LIBRARY
- FIND_PROGRAM
- FIND_PACKAGE
- EXEC_PROGRAM(bin [work_dir] ARGS <..> [OUTPUT_VARIABLE var] [RETURN_VALUE var])
- OPTION(OPTION_VAR “description string” [initial value])

References

- Software caprentry (http://software-carpentry.org)
- Dr. Frank Houdek, Michael Stupperich, Vorlesung „Management von Softwareprojekten“
- Jan Engels: Cmake Tutorial, DESY, 20th September 2007

Hanna Valtokari
Institute of Computer Science
Chair of Software Engineering
Im Neuenheimer Feld 326
69120 Heidelberg, Germany
http://se.ifi.uni-heidelberg.de
valtokari@informatik.uni-heidelberg.de

RUPRECHT-KARLS-UNIVERSITÄT HEIDELBERG