

Top 20 Tools of All Time (http://uk.gizmodo.com/)

Software Tools

Week 7 CS 212 - Spring 2008

Programming Language as a Tool

- · Use the language that best fits your task
- Think small
 - Write little programs that test various concepts
 - Test them!
 - Comment them!
 - Build collections of these little programs
 - Reuse your own code

Languages for Different Domains

- · General purpose
 - Examples: Lisp, Algol, PL/1, Scheme, Java, Python
- Systems programming
 - Emphasis on efficiency and tight control of data structures
 - Examples: C, C++, Forth, Modula-2
- Scripting
 - Examples: Unix shell, Perl, Python, Ruby, Tcl
- · Concurrent/distributed processes
 - Control of multiple threads
 - Examples: Ada, Oz, Smalltalk, Java
- Educational
 - Examples: Basic, Haskell, Pascal, Python, Scheme, Smalltalk
- · Various other domains
 - Discrete event simulation: Simula
 - · Web scripting: Javascript
 - Realtime applications: Ada
 - Text processing: Snobol, Perl
 - Printing: Postscript

Scripting Languages

- · A script is a sequence of common commands made into a single program
 - Unix uses shell scripts
 - The shell is the interactive interface to Unix
 - You can combine commands from the *Unix shell* to create programs
- A scripting language is usually
 - Easy to learn
 - Interpreted instead of compiled
- · Example scripting languages: Unix shell, Python, Perl, Tcl (Tool command language)
- · Some Python code:

class Stack (object): def __init__ (self): self.stack = [] def put (self, item): self.stack.append(item) def get (self): return self.stack.pop() def isEmpty (self): return len(self.stack) == 0

A Programming Language Controversy

- · "Go To Statement Considered
 - Edsger Dijkstra, Communications of the ACM (March 1968)
- Sparked long-running discussion on whether "go to" is necessary or desirable
 - Proponents of "go to" presented examples where code was more readable using "go to"
 - - No break
 No continue
 No exceptions
- · Led to concept of structured programming
 - . Idea: Code is clearer if we restrict ourselves to just a few control structures
 - Loops have single entry, single

Programming Language Weirdness

- Weird languages
 - Whitespace
 - . Only spaces, tabs, and newlines are significant
 - A great language for security since a program can be printed onto plain paper and stored without worrying about an adversary reading the code $\, \odot \,$
 - var'aq
 - Based on the grammatical structure of the Klingon language
- · Weird concepts
 - Polyglot code
 - Code that is valid for multiple languages
 - . Usually takes advantage of the different ways that comments are indicated in the different languages
 - Quine
 - · A program whose only output is its own source code
 - . Not considered valid to use the empty program

Integrated Development Environments

- An IDE usually includes
 - Source code editor (usually with color highlighting)
 - · Compiler or interpreter
 - Tools for "build automation" (i.e., keeps track of what needs to be recompiled)
 - Debugger
 - Class browser (for languages with classes)
- Examples: DrJava, Eclipse
 - In Eclipse: As you type, gives you list of options + documentation

- · You should know how to use a debugger!
 - Place breakpoints
 - Step through code
 - Step over
 - Step into
 - Step out of... • Examine current call-stack
 - Examine values of active variables
 - Some debuggers allow you to change a variable value
- Debuggers are usually much more effective than placing

Unix

- Original version by Ken Thompson (Bell Labs) in 1969
- operating system (not the first such system, but an early one)
- Unix is closely tied to the development of C
 - Unix was originally written in PDP-
 - 7 Assembly Language Then in B

 - Then in C
 - B and C were basically created to write Unix

- Philosophy
 - Almost everything is a text file
 - Little programs (utilities) to do little tasks
 - Connect programs with pipes & redirection

 % who | sort | lpr

 - % who | sur c
 Print an alpha
 n the active on the syste
- · Linux is an open software version of Unix
 - Since 1991
 - Linus Torvalds (the kernel)
 Richard Stallman (GNU)
 Widely used for high-performance computing
- · Mac OS X is built on Unix

Regular Expressions

- Common goal: search/match/do stuff with strings
- I dea: use special strings to match other strings
 - Some characters are meta-
- · Regular expressions are closely related to finite state automata (CS 381/481)
- Some of the rules for regular expressions
 - A regular character matches itself
 - · A . matches any character
 - * implies 0 or more occurrences (of preceding item)
 - + implies 1 or more
 - \ implies following character is treated as a regular character
 - [...] matches any one character from within the brackets; - can be used to indicate a range
- · A regular expression in Java "((\\.[0-9]+)|([0-9]+\\.[0-9]*))"

Makefiles

- · Used when compiling/recompiling a large system (many interdependent files)
 - · Checks which files have changed and only recompiles those that are necessary
 - Because of dependencies more than just the changed files can need to be recompiled
 - Also keeps track of compiler options
- · Why not recompile everything?
 - Expensive
 - Order of compilation can be important

- · Once you have a makefile
 - You recompile whatever is necessary by typing make
- · To create a makefile
 - Common strategy is to find some examples and modify them
 - There are automated tools for building makefiles
- Modern I DEs often provide tools for managing the build process

Memory Management

- Modern programs are
 - Long running
 - · Make dynamic use of memory
- · Garbage collector
 - Some languages (e.g., Java, C#) use a garbage collector to reclaim unused memory
 - Other languages (e.g., C, C++) require programmers to manage their own memory
- Manual memory management bugs
 - Dangling pointers
 - · Memory has been freed, but part of the code is still trying to use it
 - Memory leaks
 - Memory that is no longer used, but is not freed
 - Long running program ⇒ run out of memory
- . There are tools to help catch such bugs
 - E.g., purify for C, C++

Garbage Collection

- Want to keep any object that can be reached from program's variables
 - Either directly or through other objects that can be reached
 - Program's variables = anything in the call stack
- Once "not-in-use" objects are found
 - Can reclaim the memory for re-use
 - Can also compact memory
 - . I.e., move all the "in-use" objects to another memory block (without gaps between objects)

Garbage Collector Schemes

- Mark and Sweep
 - Mark every object as "not-inuse"
 - Starting from the call stack. visit every reachable object, marking it as "in-use"
 - Everything still marked "notin-use" can be reclaimed
- Reference Counting
 - Every object keeps a count of how many pointers reference
 - When count is zero, memory can be reclaimed
 - Problem: cycles!

- · For either scheme
 - Can "stop the world"
 - Can interleave (i.e., take turns)
 - · Can run concurrently
- · Java's current garbage collector
 - A 2-tier scheme (old
 - generation; new generation) A mark-and-sweep method
 - With compaction
- Java's garbage collection scheme has changed as new Java versions were released

Use of Standard Data Structures

- Packages for widely-useful data structures
 - Java Collections Framework
 - C++ STL (Standard Template Library)
 - Provide tools for
 - Sorting & searching
 - I teration
 - List • Set
 - Map (or dictionary)
 - Stack

 - Priority Queue

- · For example, Java provides
 - Interfaces
 - . List, Map, Set
 - Classes
 - ArrayList, LinkedList, HashMap, TreeMap, HashSet, TreeSet
 - Algorithms
 - Arrays.sort, Arrays.search,...

Version Control

- · Allows you to keep track of changes for a large project
 - Can back up to old version if changes create problems
 - Multiple contributors can work on the system

• SVN (Subversion) An alternative to CVS

- - Open source
 - Widely used tool for
 - · Maintains a history of all
 - Supports branching, allowing several lines of development
 - Provides mechanisms for merging branches back

• CVS (Concurrent Version System)

- version control
- changes made
- together when desired

Profiling Tools

- · People are notoriously bad at predicting the most computationally expensive parts of a program
 - Rule of thumb (Pareto Principle): 80% of the time is spent in 20% of the code
 - No use improving the code that isn't executed often
 - How do you determine where your program is spending its time?
- Part of the data produced by a profiler (Python)

ncalls	tottime	percall	cumtime	percall	filename: lineno(function)
2521	0.227	0.000	1.734	0.001	Drawing.py:102(update)
7333	0.355	0.000	0.983	0.000	Drawing.py:244(transform)
4347	0.324	0.000	4.176	0.001	Drawing.py:64(draw)
3649	0.212	0.000	1.570	0.000	Geometry.py:106(angles)
56	0.001	0.000	0.001	0.000	Geometry.py:16(init)
343160	9.818	0.000	12.759	0.000	Geometry.py:162(_determinant)
8579	0.816	0.000	13.928	0.002	Geometry.py:171(cross)
4279	0.132	0.000	0.447	0.000	Geometry.py:184(transpose)

• Java has a built-in profiler (hprof); there are many others

More Advanced Profiling

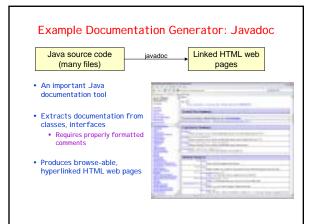
- Need additional profiling tools for applications that
 - Are multithreaded
 - Use multiple cores

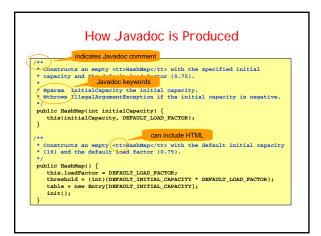


- Example: VTune Performance Analyzer (from Intel)
 - Can monitor
 - Memory usage
 - · Performance during file 1/0
 - · Thread overhead and synchronization
 - Load balancing
 - I dle time
 - Communication bottlenecks

Documentation Generators

- Comments (esp. specifications) are as important as the code itself
 - Determine successful use of code
 - Determine whether code can be maintained
 - Creation/maintenance = 1/10
- Documentation belongs in code (or as close to it as
 - "Code evolves, documentation drifts away"
 - Put specs in comments next to code when possible
 - Need to document a complicated method?
 - · Write a paragraph at the top
 - · Or break method into smaller, clearer pieces





Some Useful Javadoc Tags

@return description

- Use to describe the return value of the method, if any
- E.g., @return the sum of the two intervals

@param parameter-name description

- Describes the parameters of the method
- E.g., @param i the other interval

@author name

@deprecated reason

@see package.class#member

{@code expression}

Puts expression in code font

A List of Software Tools

- Revision control: Bazaar, Bitkeeper, Bonsai, ClearCase, CVS, Git, GNU arch, Mercurial, Monotone, PVCS, RCS, SCM, SCCS, SourceSafe, SVN, LibreSource Synchronizer
- Interface generators: Swig
- Build Tools: Make, automake, Apache Ant, SCons, Rake, Flowtracer
- Compilation and linking tools: GNU toolchain, gcc, Microsoft Visual Studio, CodeWarrior, Xcode, ICC
- Static code analysis: lint, Splint
- · Search: grep, find
- Text editors: emacs, vi
- Scripting languages: Awk, Perl, Python, REXX, Ruby, Shell, Tcl

- · Parser generators: Lex. Yacc. Parsec
- Bug Databases: gnats, Bugzilla, Trac, Atlassian Jira, LibreSource
- Debuggers: gdb, GNU Binutils, valgrind
- Memory Leaks/Corruptions Detection: dmalloc, Electric Fence, duma, Insure++,
- Purify, Aard

 Code coverage: GCT, CCover
- Source-Code Clones/Duplications Finding: CCFinder
- Refactoring Browser (e.g., Eclipse)
- Code Sharing Sites: Freshmeat, Krugle, Sourceforge, ByteMyCode, UCodit
- Source code generation tools
- Documentation generators: Doxygen help2man, POD, Javadoc, Pydoc/Epydoc



- · No hammer? No screw or screwdriver?
- Why the rifle and not the cannon? Why the watch and not the clock?
- No electricity?