Systems programming 2014-2015

Overview

- Your lecturer: Raphael 'kena' Poss
- **Course program:**
  - lectures on Tuesday 1:30pm-3:15pm (WN-M639) and Friday 9:00am-10:45am (HG-08A04), from September 2nd to October 17th inclusive.
  - Assignments the rest of the time until end of October.
- **Course materials:**
  - Course overview (this document).
  - C coding standard.
  - Quick start assessment.
  - Programming assignments (11 documents).
- **External references:**
  - Brian W. Kernighan & Dennis .M. Ritchie. *The C Programming Language*, 2nd edition. ISBN 0-13-110362-8.
  - Simon Tatham. *Descent to C*. http://www.chiark.greenend.org.uk/~sgtatham/cdescent/
  - ISO/IEC 9899:2011, *Programming Languages – C*. Final draft available online at http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1570.pdf
  - Eric S. Raymond. *The Art of Unix Programming*. http://catb.org/~esr/writings/taoup/
  - FSF. *The GNU make manual*. http://www.gnu.org/software/make/manual/
  - Bram Moolenaar. *the Vim Tutor*. `vimtutor` at the command line.
  - FSF. *A Guided Tour of Emacs*. http://www.gnu.org/software/emacs/tour/
  - Max Thayer. *How to Write a Readme Worth Reading*. http://orchestrate.io/blog/2014/07/16/how-to-write-a-readme-worth-reading/

Lecture topics

SysP from the process perspective	SysP from the system perspective
• Introduction to C • The C abstract machine model • The Unix process model • System calls and inline assembly • Address spaces and mmap() • Introduction to Rust	• fork() vs. clone() • Inter-process communication • Time and randomness • Asynchronous I/O • gdb and remote debugging • Autoconf and Automake

Assignment list

Nr	Assignment	Focus	Deadline	Duration	P-factor
1	minic	Get to know C / libraries	Sep 5th	4 days	1
2	mystr	Strings, pointers	Sep 9th	4 days	2
3	myconv	Arithmetic	Sep 12th	3 days	3
4	mystream	Buffering, syscalls	Sep 19th	7 days	4+4
5	myprintf	va_args	Sep 22nd	3 days	3+2
.	myheap*	Virtual memory	Sep 28th	6 days	4+8
6	myls	File system, stat, options	Oct 3rd	5 days	4+3
7	sigcopy	Signals and side channels	Oct 10th	5 days	4+4
8	mytftpd	Network sockets, time	Oct 17th	5 days	4+4
9	cryptserv	Unix sockets, select	Oct 24th	5 days	4+6
.	minishell*	Process control	Oct 31th	10 days	8+12

The assignments whose name is followed by an asterisk are "bonus" assignments, see Grading below.

All assignments must adhere to the accompanying **C coding standard**. *Read it before starting work on your assignments.* Each submitted archive must be named after the name of the assignment in this table (eg. `minic.tar.gz`).

Programming environment

Editor: Emacs or vim strongly recommended. Think about learning something new vs. long-term benefit of using a stable tool. **OS for coding:** Beware of line endings and file name cases. **Portability of C code:** Beware of differences between BSD (incl. OSX) and SysV (incl. Linux) regarding the interaction between signals and syscalls, and wrt. the persistence of signal handlers.

Grading

The grading for this course uses two sources for credits:

- the 9 mandatory programming assignments, which determine the bulk of the final grade;
- the 2 bonus assignments, which earn a bonus percentage;
- class participation, which can earn a fixed bonus of up to 0.5 point on the final grade.

The formula to compute the final grade is as follows:

1. at the end of the period, a "base grade" *B* is computed as follows:
   - if the non-weighted sum of the mandatory assignment grades is greater than or equal to 42, then the base grade is 6.
   - otherwise, the base grade is equal to the non-weighted sum of the mandatory assignment grades divided by 7.

   In other words, $B = \frac{\min(42, \sum_{i=1}^{9} C_i)}{7}$
2. The extra credits above 6 for every mandatory assignment are then weighted by the assignment durations to compute an "extra grade" *E*, rounded down to the nearest multiple of 0.5.
   In other words, $E = \frac{1}{2} \lfloor 2 \frac{\sum_{i=1}^{9} d_i \times \max(0, C_i - 6)}{\sum_{i=1}^{9} d_i} \rfloor$
3. The "intermediate" grade *I* is equal to either *B* if *B<6*, or *B+E* if *B* is 6 or greater, rounded to the nearest multiple of 0.5.
4. The two bonus assignments (myheap and minishell) then determine a factor *K* as follows: *K* starts at 1, and is incremented by 0.05 (+5%) for every 2 points you score on the bonus assignments.
   In other words, $K = 1 + 0.05 \times \lfloor \frac{(C_{\text{myheap}} + C_{\text{minishell}})}{2} \rfloor$
5. The final grade *F* is the intermediate grade *I* scaled by *K*, ie. $F = I \times K$. The 0.5 bonus, if earned, is then added to this. Of course, the final grade is also capped to 10.

Some examples:

- Bart failed 3 assignments entirely (grade = 1) but obtained exactly 6 to all other assignments. B = 6, E = 0; Bart gets a final grade between 6 and 6.5.
- Denise failed the first 3 assigments entirely (grade = 1) but obtained 10 to all other assignments. B = 6, E = 3; Denise gets a final grade between 9 and 9.5.

Overall this grading system is more generous than a simple weighted average rounded down. It allows a student to completely fail up to 3 assignments and still get a good grade, as long as they perform above average on the remaining assignments.

Work organization and P-factor

The rules for this course are simple: read, write code, check/test. **Hard.**

But how hard is "hard"? This question is particularly relevant for students who think they already know how to program well, and therefore feel entitled to work less (or less hard). Is it "hard" as in "I must attend all the sessions and spend 2-4 hours every day after school to study"-hard? Or "I can start coding one hour before the deadline and still get a passing grade"-hard?

There are two answers depending on your work ethics.

If you know that you will invest at least 2 serious study hours every day, and allocate the time needed to read supplementary materials whenever you can (eg. on the bus, before sleep, etc.), then you can do just that, and it should work. You will even probably get a good grade.

If you know about yourself that you have a procrastination problem (NB: most students in CS have one, be honest with yourself), then you should use the P-factor in the table above.

The P-factor works as follows.

First, think about how procrastination works. Imagine you have a deadline by next Monday morning. During the week, you casually read the assignment text and estimate it should be easy enough to do during the week-end. Maybe you will quickly scan some articles or a tutorial on the topic. On Saturday morning, your family organizes an outing, so you think you will work on Saturday night. But then your friends ask you out on Saturday night, so you decide to work long hours on Sunday. On Sunday morning, you turn your computer on, start the code editor, and try to get started with the assignment. But then you open one web page, then another, read plenty of ancillary articles that are only remotely related to the assignment, but which you find interesting anyway. Before you realize, it is already dinner time and time for news on TV or your favorite sitcom. And then the stress starts becoming unbeareable, so you sit at the computer and actually get going *hard* with the assignment.

If the P-factor has a single number X, and your Sunday evening frenzy starts at least *X* hours before the deadline, then you *may* have a chance to complete the assignment so as to get a passing grade. It *will* require that amount of "rush hours", though: high-stress, high-energy, lots of caffeine, bad sleep. If you have ever procrastinated on something important, you know what I mean already. It also only works if you are already sufficiently proficient (A2-B1 according to this table).

If the P-factor has two numbers "Y+X", then you are absolutely screwed. You will *not* be able to finish the assignment, even if you have *X+Y* late night hours available before the deadline.

The form "Y+X" means that you must spend at least a P-factor Y worth of effort on the assignment first, then *sleep a full night over it*, then spend another P-factor X worth of effort. The "sleep break" is necessary to mentally check your work and realize how to finish/test/improve it.

But really, don't procrastinate. The high-stress, high-energy effort necessary for this kind of "final rush" is extremely unhealthy (speaking from experience), and will not get you grades as good as if you worked regularly.

Copyright and licensing