

HW #8

CSC230 Section 001 Fall 2007

Due Date: Wednesday, December 12, 11:45pm EST

NOTE: THIS HOMEWORK IS EXTRA CREDIT! DO ANY OR ALL OF IT – POINTS YOU RECEIVE WILL BE ADDED TO YOUR HOMEWORK SCORE. IF YOU ARE SATISFIED WITH YOUR CURRENT HOMEWORK SCORE, OUT OF 700 POINTS, YOU ARE NOT REQUIRED TO SUBMIT THIS ASSIGNMENT FOR CREDIT. IT IS EXPECTED, HOWEVER, THAT YOU KNOW THE CONTENT OF THIS HOMEWORK AND CAN ANSWER THE QUESTIONS BELOW, AND WRITE THE EQUIVALENT PROGRAMS.

Purpose

To give you practice in distributed, concurrent, and secure programming, as well as in using tools like subversion and make.

Group or Individual Work

Homeworks may be worked on in groups of 2. Submit one homework, under only one group member's name, but with both members identified in the source code. If you prefer to work alone, that is allowed.

Getting Help

Come to my office hours, ask questions after class, go to the TA office hours, use the message board, or send email; all are good ways to get help.

Programming Requirements

As before, your programs need to compile with `gcc -Wall -std=c99` and execute properly on the Linux OS+Intel PC platform.

Programs must be formatted cleanly and consistently.

Commenting requirements are the same as before (see hw1).

Always return 0 upon successful completion of execution of your program, and something other than 0 otherwise.

You do not have to check for invalid inputs. Only valid inputs will be tested with your programs.

The Problems

pgm36

Choose one of the following. Whichever one you choose, use subversion to maintain a source code repository. Ask us to set up the subversion access and repository for you. Submit with your code whatever files you want, with whatever names you want, and a Makefile named **Makefile36**. The Makefile should have targets:

pgm36client - make the executable for the client program

pgm36serv - make the executable for the server

clean - remove all non-source-code files from the current directory

Either:

Write a distributed poker game. The server shuffles and deals cards, and keeps track of betting. The clients request cards and place bets, on behalf of a player, and keep track of their balance, and show what cards the other players have. The player who first starts the game determines who wins each hand.

or:

Write a distributed appointments calendar program. The server keeps track of appointments for a user, displays the appointment calendar for a day on request, sends reminders 15 minutes before appointments, and prevents appointment conflicts. The client sends appointment updates, and can request to see the calendar. Then, update the server so it keeps track of appointments for a set of users, and allows users to query each other's calendars, and make appointments with other users (updating both of their calendars).

or:

Enhance the jokeserv/jokeask programs to allow each client to stay connected and ask as many questions as they like, for as long as they like. Your server will need to be concurrent so that multiple clients can be connected at the same time. Also enhance the program so that users can supply new jokes to the server, and so that users can request random jokes be told by the server.

pgm37a.c and pgm37b.c

Version a:

Write a program that creates a child process to read in a sequence of numbers, one per line, from the standard input. When the child process finishes reading the input, it outputs (to standard output) the sum of the numbers, terminates execution, following which the parent process outputs **"Child process has finished\n"**.

Version b:

Rather than the child process outputting the sum, it communicates the sum back to the parent process, which outputs the sum, and then outputs **"Child process has finished"**.

hw8.txt (pencil and paper problems)

1. Given the following Makefile, what commands will be executed when **make** is executed if the file modification times of all .c and .h files are 8:15am, and the file modification times of all other files (including all of the object

files and the executable) are 9:15am, except for:

* `main.c`, which has a modification time of 10:15am, and

* `compare.o`, which has a modification time of 7:15am

```
BASE    = /home/barney/progs
CC      = gcc
CFLAGS  = -O -Wall -std=C99
EFILE   = $(BASE)/bin/compare_sorts
INCLS   = -I$(LOC)/include
LIBS    = $(LOC)/lib/g_lib.a $(LOC)/lib/h_lib.a
LOC     = /usr/local
OBJS    = main.o another_qsort.o chk_order.o \
          compare.o quicksort.o
$(EFILE): $(OBJS)
    @echo "linking..."
    @$ (CC) $(CFLAGS) -o $@ $(OBJS) $(LIBS)
$(OBJS): compare_sorts.h
    $(CC) $(CFLAGS) $(INCLS) -c $*.c
clean:
    rm *~ $(OBJS)
```

2. Modify the above Makefile so that:

1. only `main.o` and `compare.o` depend on the header file
2. the `compare.c` and `quicksort.c` files will be found in directory `$(LOC)/src`. All the `.o` files and the other `.c` files will be in the current working directory, however.
3. the target "run" will execute the program with no command line options, and with the standard input redirected from file `../inputs/in1.txt`

3. In the following code, spot the bug(s) and explain how it (they) can be exploited by a hacker, and how to correct the bugs:

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

char filenm[100];
char c;

int main(int argc, char *argv[]) {
    (void) strcpy(filenm, argv[1]);
    FILE * infile = fopen(filenm, "r");
    while ((c = getc(infile)) != EOF)
        putchar(toupper(c));
    return 0;
}
```

4. Write a program that has the following variable declarations:

```
double q = 3.1;
double r = 2.5;
int a[5] = {0, 0, 0, 0, 0};
int b = 5;
int c = 20;
```

1. In your program, make these global variables. Write a loop that modifies the values in **a**. For what values of the loop index will the values of **b** and **c** be changed? For what values of the loop index will the values of **q** and **r** be changed?
2. Answer the same questions as above, but make the variables **q**, **r**, **a**, **b**, and **c** local (auto) variables of a function which is called from **main()**. Then answer the same question as above (i.e., "For what values..."). Is there any difference?

Show your program in both cases.

5. A programmer wishing to write an efficient program dynamically allocates memory for a data structure, using **malloc()**. Her program stores some secret data in this data structure. At some point in her program, this data structure is no longer needed, but space for another data structure of the same size or smaller is needed. Which is better, and why?

- **free** the memory for the first data structure, then **malloc** memory for the second data structure
- scrub the memory for the first data structure, and reuse that same area of memory for the second data structure
- **realloc** the memory for the first data structure to the size needed for the second data structure

6. Suppose a process forks off a child, which shares the standard input and output with the parent. The parent reads a line from standard input and prints "**parent: <line goes here>**", where **<line goes here>** is the line it just read. The child does the same, but prints "**child: <line goes here>**". If the standard input is

```
One
Two
Three
```

What are all the possible outputs you could see? Example:

```
parent: One
parent: Two
child: Three
```

Testing

Sorry, no reference implementations will be provided. Solutions will be posted, however.

Instructions for Submission

Submit whatever files you want for pgm36, including Makefile36.
Also submit files pgm37a.c and pgm37b.c and hw8.txt.

Make sure your programs compile on Linux OS+Intel PCs cleanly, using the options given!

Grading

- 45 points for successfully implementing pgm36
- 15 points for successfully implementing pgm37a and pgm37b
- 40 points for answering questions correctly in hw8.txt