CS 258, Midterm Extras, Winter 2015

Terms and Conditions. The solutions are due by class time on Thursday February 19. All conditions from the main midterm apply.

Note: In the following problems, you are **not** allowed to use `root` privileges. Your solution should only rely on writing, compiling, and running code with regular used privileges. Further, in your solution you are not allowed to attach any debugger to running processes. (You may, of course, use a debugger or tracer as much as you like to debug your solution.)

Problem 1. *Ouch! It traps!*

A program below will attempt to access an address outside of its address space. Allow this access to be executed as written; it will cause a memory protection trap. Write a signal handler to allow the program to survive this trap and finish with the final `printf()`!

You can modify the address or the program, but only *after* the fault occurs. Note that the fault is a trap, that is, the instruction that caused it will be re-started after the signal handler code runs.

```
/* Build me with gcc -m64 */
#include <stdio.h>
#include <signal.h>
#include <stdlib.h>
#include <errno.h>

int main(){
  const unsigned long long badaddr = 0xffffffffbadfeed;
  char *p;

  /*
  Set up signal handling.
  */

  OMGGPF:
  p = (char*) badaddr;
  *p = '\0';

  printf("Phew! Survived a general protection fault!\n");
  return(0);
}
```

Problem 2. *Lie to me.*

A program is very particular about the user ID under which it runs: it checks it and refuses to run if it isn't just right. Unfortunately for you, your user ID is not that user, and you only get the binary of the program, and you can't even edit the binary.

You could, of course, connect a debugger to this program and manually bypass the check, but you want to be able to run it simply and conveniently. Luckily for you, this program is dynamically linked.

Write the code and produce the shell command that will run this binary for you, without modifications.

```
#include <unistd.h>
#include <sys/types.h>
#include <stdio.h>

int main(){

  /* ... */

  if( getuid() != 100 ){
    fprintf( stderr, "Not the right user! Exiting.\n");
    exit(-1);
  }

  printf("Welcome, correct user, let's do some work\n");

  /* ... */

  return(0);
}
```