CSE306 Software Quality in Practice

Dr. Carl Alphonse
alphonce@buffalo.edu
343 Davis Hall
PRE

- Document baseline approach to SW development in a team environment
- What are we looking for?
  Documentation of process.
- Some teams did not collaborate/communicate well.
  Something to work on: how can you (as an individual & as a team) encourage/ensure collaboration and communication?
Learning outcomes of course

(I) Employ static and dynamic analysis tools to detect faults in a given piece of software.

(II) Employ profiling tools to identify performance issues (both time and memory) in a given piece of software.

(III) Employ testing frameworks to write tests that fail in the presence of software faults, and pass otherwise.

(IV) Employ a structured, methodical approach to detecting, testing, identifying and correcting software faults.

(V) Work productively as a member of a software development team.
Think broadly
Think broadly

build to LPR
Think broadly

build to LPR

apply in other courses
Think broadly

build to LPR

apply in other courses

showcase to potential employers
Released later today
Team-based: same teams as for PRE
Clone repo via GitHub as usual so course staff can view

**Learning goals:**
- show you can apply process
- show you can use tools effectively
- show you can engage in teamwork
  - communication and collaboration are key
- More to come between EXP01 and EXP02
More gdb commands

- C-x C-a toggle between a "graphical" and line-based UI
- break <line> (e.g. break 31)
- info b (list breakpoints)
- c (continue to next breakpoint), c 10
- watch <variable> (e.g. watch i)
  - [link](https://sourceware.org/gdb/current/onlinedocs/gdb/Set-Watchpoints.html#Set-Watchpoints)

Looking at source code:
- list line#
- list function
- disassemble /m

Looking at data:
- print
- examine (x)
  - x /s name, x/48c name (addresses in hex!)
- [link](https://sourceware.org/gdb/current/onlinedocs/gdb/Memory.html#Memory)
```c
#include <stdlib.h>
#include <string.h>
#include <stdio.h>

int main(int argc, char * argv[]) {
    if (argc != 2) {
        printf("Please give one numeric argument.\n");
        return 1;
    }

    int limit = atoi(argv[1]);

    char * string, * name;
    name = malloc(3 * sizeof(*name));
    string = malloc(9 * sizeof(*string));
    name[0] = '@';
    name[1] = '$';
    name[2] = '\0';
    string[0] = 's';
    string[1] = 'e';
    string[2] = 'r';
    string[3] = 'e';
    string[4] = 'n';
    string[5] = 'i';
    string[6] = 't';
    string[7] = 'y';
    string[8] = '\0';
    printf("string has length %zu and is %s.\n", strlen(string), string);
    printf("name has length %zu and is %s.\n", strlen(name), name);
    for (int i=3; i<limit; i++) {
        name[i] = (char) ('a'+((i-3)%26));
    }
    name[limit] = '\0';
    printf("string has length %zu and is %s.\n", strlen(string), string);
    printf("name has length %zu and is %s.\n", strlen(name), name);
    return 0;
}
```
```c
#include <stdlib.h>
#include <string.h>
#include <stdio.h>

int main(int argc, char * argv[]) {
    if (argc !=2) {
        printf("Please give one numeric argument.\n");
        return 1;
    }

    int limit = atoi(argv[1]);
    char * string,* name;
    name = malloc(3 * sizeof(*name));
    string = malloc(9 * sizeof(*string));
    name[0] = '@';
    name[1] = '$';
    name[2] = '\0';
    string[0] = 's';
    string[1] = 'e';
    string[2] = 'r';
    string[3] = 'e';
    string[4] = 'n';
    string[5] = 'i';
    string[6] = 't';
    string[7] = 'y';
    string[8] = '\0';
    printf("string has length %zu and is %s.\n",strlen(string),string);
    printf("name has length %zu and is %s.\n",strlen(name),name);
    for (int i=3; i<limit; i++) {
        name[i] = (char) ('a'+((i-3)%26));
    }
    name[limit] = '\0';
    printf("string has length %zu and is %s.\n",strlen(string),string);
    printf("name has length %zu and is %s.\n",strlen(name),name);
    return 0;
}
```