Operating Systems

Project 1 Feedback

Grading Guidelines

• 10% major bug
  – segfault
  – memory leak
  – major feature not working

• 5% minor bug
• 2-3% cosmetic bug
• Less deduction if you know about a bug, even if you couldn’t fix it

• 4.0 >= 95%
• 3.0 >= 75%
• 2.0 >= 50%

README

• Helpful information for granting credit
  – Known bugs
  – Extra credits: clearly state
    • Not implemented
    • If partially implemented, which parts
    • Fully implemented
  – Additional features

• No need to include standard stuff
  – i.e. type make to compile

• “How to” sections are only necessary if your program compiles or runs differently from spec – it shouldn’t happen without a good reason

Avoid Circular Includes

• Always enclose your .h with these directives:
  #ifndef NAME_H
  #define NAME_H
  /* contents */
  #endif

Avoid Unnecessary printf

• Sometimes it’s necessary to inform the user
• Otherwise, have a way to disable all debugging output in the finished product

#define (DEBUG)
#endif

//printfs
#endif

$(CC) $(CFLAGS) -DDEBUG -c hw1.c

No Integer Constants

• #define them
  #define FALSE 0
  #define TRUE 1
  #define SIZE 100

• use enum
  enum{FALSE,TRUE};
  enum{BLACK,RED,GREEN,BLUE};

• use #defines
  _NSIG //signal.h
  STDIN_FILENO
  STDOUT_FILENO
  STDOUT_FILENO
Things to Note

- Compile with `-Wall`
- Fix the warnings, do not ignore them
- Your Makefile should have the following targets:
  - `make` (compile to default target)
  - `make clean` (removes all .o and executable)
- Comment your globals
- Comment your function parameters
- Do not submit deep directory structures
  - call the "submit" script one single level up from where your code is

Arrays/Lists Heap/Stack

- Use arrays unless you don't know the size ahead of time and it changes constantly
- `malloc` an array if size is only known at run time
- Keep to automatic storage if you can
  - structs, arrays
- Automatic storage lives on the stack
- Dynamic storage lives on the heap
- Dynamic storage must be freed

Arrays/Lists Heap/Stack

- int arr[n];
- int *arr = malloc(sizeof(int)*n);
- char *str;
- char str[6];
- char *str = "CS355";
- char str[] = "CS355";
- char *str = malloc(sizeof(char)*6);

Memory and Pointers

- char *str;
- char str[6];
- char *str = "CS355";
- char str[] = "CS355";
- char *str = malloc(sizeof(char)*6);

hw1 specifics

- `./hwlb -s -l|0|a`
- `ctrl-d` sends EOF
  - `STDIN` terminates on `ctrl-d`
- signal handler should have been registered to ALL signals, not just some arbitrary ones
  - use a loop, and `_NSIG`

hw1 specifics

- char buff[SIZE];
- `fgets/read` - use SIZE-1
- `fscanf("intr %d", &interrupts);
- open(outfd, O_WRONLY|O_CREAT|O_TRUNC, 0666);
- creat
- umask(S_IWGRP|S_IWOTH); // 022

Problems with lab machines

- Generate tickets at: systems.cs.brynmawr.edu
- If urgent, send email to ianj@cs.brynmawr.edu
- Or run to his office if you fork-bombed
- Telling Ian is more effective than telling me (I have to go to him anyway)
Source Code Organization

• Split into multiple source files
  – ll.c and hw1.c
• Every .c should have its own .h
• Might be useful to have an additional common.h

• Any operation on your data structure (i.e. linked list) outside of ll.c should be done via a function call provided through ll.h