15-213:
Introduction to Computer Systems

It's “fun”

Buffer Lab, Shell Lab, Errors, and Coding Style

Recitation n
Monday October 5th, 2009

Slides by: Hunter Pitelka

Carnegie Mellon
School of Computer Science

Schedule

- Last Minute Bufferlab questions
- Quick intro to shell lab
- Correctly handling Error Conditions
- Code Style

Bufferlab

- Due Tomorrow (10/6) at 11:59:59pm to autolab.
- The grade autolab shows you is NOT your actual final grade.
  - We will go over your submissions by hand and verify they are correct
  - Just follow what the writeup says, and you'll get full points.
  - Do not jump past any stage verifications, or just straight to the validate function!

Final (quick) Bufferlab Questions?

System Call Error Handling

- Sometimes things go wrong.
  - In the OS
  - On Disk
  - Across the Network.
- You will be told about these errors though a system call failing.
- You MUST handle these errors in a “correct” manner.
  - What does correct mean?

System Call Error Handling

- How do we handle errors?
  - Notify the user.
  - Try again.
  - Try something different.
  - Abort.
- What you do depends on what happened.
  - No correct answer for every error condition.
  - Consult man pages to educate your decision.

System Call Error Handling

fork() man page

```
RETURN VALUE
       On  success,  the  PID  of  the  child process is returned in the
parent, and 0 is returned in the child.  On failure, -1 is returned in the
parent, no child process is created, and errno is set appropriately.

ERRORS
       ENOMEM fork() failed to allocate the necessary kernel structures
because memory is tight.
```

System Call Error Handling

```
RETURN VALUE
       On  success,  the  number of bytes read is returned (zero indicates
end of file), and the file position is advanced by this number.  It is not an
error if this number is smaller than the number of bytes requested....  On
error, -1 is returned, and errno is set appropriately.

ERRORS
       EAGAIN Non-blocking I/O has been selected using O_NONBLOCK and no data
was immediately available for reading.

       EBADF  fd is not a valid file descriptor or is not open for reading.

       EFAULT buf is outside your accessible address space.

       EIO    I/O  error. This  will happen for example when the process is
in a background process group, tries to read from its controlling tty, and
either it is ignoring or blocking SIGTTIN or its process group is orphaned.
It may also occur when there is a low-level I/O  error  while
              reading from a disk or tape.

       EISDIR fd refers to a directory.
```

System Call Error Handling

- Can you recover from the error and continue execution?
  - Then do it!
  - Inform the user what went wrong, ask them to fix it if possible, and continue running.
- There is no way to continue execution
  - Inform the user why you are exiting
  - Clean up
  - Exit gracefully.

Good Code Style

Good Code Style = readability

- Commenting
- Use of whitespace
- Variable names
- Straightforward logic. (not always possible)

Commenting

- Comments should explain your code, and your thinking.
- Comments should NEVER contain code!!!
  - //printf(“debug value is %d\n”,value);
  - /*This does (x ^ y) ^ z*/
- If your code doesn't need a comment, then don't comment it!
  - var = x + y; /*this stores x +y into var*/

Commenting

- As a general rule, you should explain every
  - File
  - Function
  - loop/conditional
  - Confusing block of code. (aka Algorithm)
- If your comments do not add to the readability of code, then take them out.
- Humor is okay, obscenities are (mostly) not okay.

Use of whitespace

- In general, whitespace increases readability.

```
int replaceByte(int x, int n, int c) {
    int bytePos = n << 3;
    int replacement = c << index;
    int byteMask = (0xFF) << index;

    mask = ~mask;

    return (byteMask & x) | replacement;
}
```

```
int replaceByte(int x, int n, int c) {
int bytePos = n << 3;
int replacement = c << index;
int byteMask = (0xFF) << index;
mask = ~mask;
return (byteMask & x) | replacement;
}
```

Use of Whitespace

- All code and comments in a block of code should be indented to the same level
  - A “block” is everything inside a function, loop, conditional, or otherwise separated code.

```
void fun(arg_t arg){
/*this function rocks*/
   Var = statement;
   Var = awesomeStatement;
/*now for something completely
different*/
   Var3=coolStatement;
   Var4= leetHax;
}
```

```
void fun(arg_t arg){
   /*this function rocks!*/
   Var = statement;
   Var = awesomeStatement;
   /*now for something completely
     different*/
   Var3=coolStatement;
   Var4= leetHax;
}
```

Use of Whitespace

- Tabs vs. Spaces
  - Lets not go there.
  - Just be CONSISTENT!!!!!
    - Either, always use tabs, or always use spaces!
- Excessive Whitespace
  - More than one blank line is usually necessary
- Whitespace & Comments
  - Aligning comments is acceptable, but either do it always, or don't do it at all.

Variable Names

- Should be descriptive
- Should explain what is being stored in the variable
- Do not use single letter variable names
  - i,j,k are okay for loop iterators
- Use either camelCase or under_scores.
  - Pick one and stick with it!

Straightforward Logic

- Your code should be easy to read and easy to understand.

```
if(a)
  if(b) x=y;
else x=z;
```

```
char *p;
switch (n)
{
case 1:
    p = "one";
    if (0)
case 2:
    p = "two";
    if (0)
case 3:
    p = "three";
    printf("%s", p);
    break;
}
```

Braces

- {'s and }'s should always be used:

```
if(a)
  if(b) x=y;
else x=z;
```

- It doesn't matter what line they are placed on, just use them, and be consistent.

```
if(a){
  if(b){
    x=y;
  }else{
    x=z;
  }
}
```

```
if(a)
{
  if(b)
  {
    x=y;
  }
  else
  {
    x=z;
  }
}
```

Questions?

```
char * answerStyleQuestion(char * input){
    if(strcmp(input,”is this good style?”) ==0){
       return “what do you think?”
    }else{
       return pokeTA(input);
    }
}
```

tshlab

- Write a shell for unix.
  - Process Control
  - I/O Redirection
- Read the handout, and READ MAN PAGES!!
  - You will need to know many details about how system functions work in order to write this lab.
- An emphasis on interaction with the system.

kthxbai