

CSE2031 Software Tools - Testing and C (cont.)

Summer 2010

Przemyslaw Pawluk

Department of Computer Science and Engineering York University Toronto

May 11, 2010

Notes			



## What we did last time

#### Overview of C

- Introduction to the language
- Program structure
- Types in C
- Operators in C
- IO and Files in C

### Overview of UNIX

- Why UNIX?
- Philosophy of UNIX
- Structure of UNIX

Notes			



## What we will do today?

1	Introd	uction	to	Lests

- Random tests
- Black-box tests Glass-box tests
- Regression tests
- Boundary conditions testing
- Pre- and Post-condition testing
- Assertions
- Example
- 2 C-continuation
  - Functions
  - Scope
  - Preprocessor

Notes			



# Why Tests?

- 1990 AT&T long distance calls fail for 9 hours
  - Wrong location for C break statement
- 1996 Ariane rocket explodes on launch
  - Overflow converting 64-bit float to 16-bit integer
- 1999 Mars Climate Orbiter crashes on Mars
  - Missing conversion of English units to metric units
- Therac: A radiation therapy machine that delivered massive amount of radiations killing at leaset 5 people
  - Among many others, the reuse of software written for a machine with hardware interlock. Therac did not have hardware interlock.

ivotes		



## Idea of testing

• Testing is getting sure your code is correct (no bugs).

- In reality, you can only detect the existence of bugs, not their absence (Dijkstra).
- Multiple runs of code using different inputs.

Notes			



### Unit tests

• Do not wait till you complete the program to test it, test every piece that you write (function, block, if, )

- If you wait until something breaks, you probably have forgotten what the code does.
- It takes time because sometimes additional work has to be done i.e. stub



## Testing

What do you need for testing?

- The code you want to test
- Some inputs
- What is the "correct" output of the above inputs, so you can compare.

Test Coverage

Did you cover every statement in the code?

Notes			



### Random input

- Random inputs to the program
- Easy to do
- Without a statistical framework, the results are meaningless.

Notes			



## Black-box

• No knowledge of the implementation (code)

- Test based on the specifications.
- Tests prepared before implementation.
- Tests prepared by some one else other than the person who will write (wrote) the code.
- May not test every path in the program!



## Glass-box

```
• Full knowledge of the program.
```

• Test cases should test (cover) all different paths in the program.

```
\quad \textbf{if(} \ a > \ b \, ) \ \{
       \quad \text{if} ( \ c>\!\!=\!\!\! d) \ \{
              y = \dots;
               else \ \{
       else {
```

```
Notes
```



## Over and over again

•	When you	fix a	bug.	VOII	mav	introduce	another	bug.

- When you fix a bug, you may break another fix
- When you create a test, keep it
- When you fix a bug, apply all previous tests

Notes			



# Boundary conditions testing

Example What is the boundary condition? How can we identify boundary conditions?

Notes		



# Pre- and Post-conditions

Preconditions Check if the input is correct

Postconditions

Check if your output is correct

Notes			



## Assertions

- You can use assertion facilities in <assert.h>
- Use it only when the failure is really unexpected and there is no way to recover (however you may use it to test pre-, post-conditions and loops' invariants)
- assert (n>0); If that is not true, the program terminates with a message  $% \left( 1\right) =\left( 1\right) \left( 1\right) \left$ saying the assertion failed.

Notes			



# Example

Lets consider the GCD

Notes			



# Functions and Scope

Functions

- Brake large computing tasks into smaller
- Can be reused

Scope

Where the name can be used/visible?

Notes



## Function – basics

Limitations of human perception

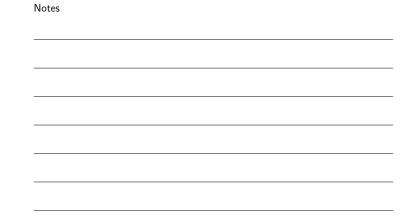
Human usually can focus on 5+/-2 elements

Brake complex tasks into smaller

During the design stage try to separate small tasks that may be implemented as single function.

Simple rule

Try to fit the function on one screen





### Definition and Declaration

Declaration returned\_type function\_name(list\_of\_arguments)

Definition

returned\_type function\_name(list\_of\_arguments) declarations and statements

Return statement

return expression;



CSE2031 Software Tools -Testing

Przemysk

ntroduction Tests
Random tests
Black-box tests
Black-box tests
Regression tests
Boundary conditions testing
Pre- and
Post-condition testing
Assertions

Ccontinuatio Functions Scope Prædrokson ".c", ".h", ".o'

#### Header ".h"

- System header files declare the interfaces to parts of the operating system.
- Your own header files contain declarations for interfaces between the source files of your program.
- Including a header file produces the same results as copying the header file into each source file that needs it.
- In C, header files names end with .h. It is most portable to use only letters, digits, dashes, and underscores in header file names, and at most one dot.

### Body ".c"

Contains includes of headers and definitions of functions.

#### Object ".o"

Object files are compiled from the source+header files. We can have program divided into several "modules". All modules then can be compiled separately and linked later into the one executable.




#### Return

Software Tools -Testing and C (cont.)

Przemys Pawlul

Introduct to Tests Random tests Black-box tests Glass-box tests Regressior tests Boundary conditions testing Pre- and Postcondition testing Assertions Example

Pr**2**27/04

- Function uses *return* statement to return the result to the
- Functions can return arbitrary type: void, int, double, pointer (to the variable or function) etc
- Inconsistent expression will be casted to the returned\_type of the function

Notes			



### External variables

CSE2031 Software Tools -Testing and C (cont.)

Przemys

Introduction to Tests
Random tests
Black-box tests
Glass-box tests
Regression tests
Boundary conditions testing
Pre- and
Post-condition testing

Internal variables

Defined inside of the function body and exists only when the function is executed  $% \left( 1\right) =\left( 1\right) \left( 1\right)$ 

#### External objects

- External variables and function are defined outside of any function.
- External variables may be used as a tool to communicate between functions



#### Too many externals is not good

CSE2031 Software Tools -Testing and C

Przemysla

ntroduction Tests
Random tests
Black-box tests
Glass-box tests
Regression tests
Boundary conditions testing

Continuation
Functions
Scope
Protrictions

Important!

Do not misuse external definitions (global variables)!

Problem with externals

- Everyone can access the variable (like *public* member in Java)
- Low level of control
- Too many externals leads to bad program structure with too many data connections between functions (problem with reusing)

Notes			



### Scope - Definition

Software Tools -Testing and C (cont.)

Przemysla

Introduction to Tests

Random tests

Black-box tests

Glass-box tests

Glass-box tests

Boundary conditions testing

Pre- and

Postcondition testing

Assertions

Example

Ccontinuat Functions Scope Following questions should be answered:

- How to write declarations so that variables are properly declared during compilation?
- How are declarations arranged so that all the pieces will be properly connected when program is loaded?
- How are declarations organized so there is only one copy?
- How external variables are initialized (so that all of them are initialize once)?

### Scope – Definition

Scope is a part of the program within which declared name can be used  $% \left( 1\right) =\left( 1\right) \left( 1$ 



Notes

Notes



### Extern declaration

CSE2031 Software Tools -Testing and C (cont.)

Przemysl

to Tests
Random tests
Black-box tests
Glass-box tests
Regression tests
Boundary conditions testing
Pre- and Postcondition testing
Assertions

Ccontinuations If we want to use the variable before it's definition or in other file then  ${\tt extern}$  declaration is required.

file1.c

extern int size;
extern char buf[];

file2.c

int size = SIZE;
char buf[SIZE];



# Static Variables

Declaration static allows us to restrict the visibility (scope) of the variable (hide).

file1.c

static int size = SIZE; static int size = SIZE; static char buf[SIZE];

file2.c static char buf[SIZE]; Notes



# Static variables – Example

Notes

Notes



# Register

Just a suggestion

 $\label{eq:Declaration} \mbox{Declaration register is a suggestion or advice addressed to the}$ compiler that the variable will be often used and should be placed in the machine register (fast access).

# Restrictions

- Size of registers
- Number of registers
- Size of type



### Blocks and Declarations

```
• Variables can be defined in the beginning of the block:
```

- function
- $\bullet$  compound statement (just after the left brace  $\{)$
- inner declaration hides outer declaration
- automatic variables (i.e. parameters) hides external variables

```
\quad \textbf{int} \quad \mathsf{i=}1;
if (i>0)
          int i;
          \quad \  \text{for} \, (\,\, i = \! 0; i \! < \! \! \text{MAX}; \, i \! + \! + \! )
          {
```

```
Notes
```



#### Initialization

In absence of explicit initialization:

- external and static variables are initialized to be zero;
- automatic and register variables have undefined value (garbage!!!)

How to do it?

- external and static needs constant expression, done once before program runs;
- automatic and register initializer not restricted, done each time the block is entered;
- arrays are initialized by the list of members

```
int x = 0;
char txt2[] = { 't', 'e', 'x', 't', '\0' };
```

Calling the function by itself directly or indirectly

Notes			



### Recursion

Definition

/*     * recursive GCD     */ int gcd(int m, int n) {         /* base case(s) m or n equals 0*/         if (m == 0)             return(n);         if (n == 0)
<pre>return(m);  /* now recurse */ return(gcd(n, m % n)); }</pre>

Notes		



## Recursion – Example

CSE2031 Software Tools -Testing and C

Przemysl

Introducti to Tests Random tests Black-box tests Glass-box tests Regression tests Boundary conditions testing Pre- and Postcondition testing Assertions

Functions Scope

Notes			



#### Include

Software Tools -Testing and C (cont.)

rzemys

Introducti to Tests Random tests Black-box tests Glass-box tests Boundary conditions testing Pre- and Postcondition testing Assertions Example #include "file" or #include <file>

- includes content of file during the compilation
- when file is quoted ("") searching for the file begins in the dir where source program is
- if it is not found there or it is surrounded by "<" and ">" implementation defined rule is used to find included file (in in specified directory)
- includes may be cascade (included file may contain another includes)
- when included file changes all dependent source files (that have included it) should be recompiled

Notes			



# Define

CSE2031 Software Tools -Testing and C (cont.)

Hov

Przemysi Pawluł

Introducti to Tests Random tests Black-box tests Glass-box tests Boundary conditions testing Pre- and Postcondition testing

Ccontinuat Functions Scope #define name replacement\_text

How it works?

Subsequent occurrences of name will be replaced by the replacement\_text.

How to build it?

- Name has the same form as variable name
- replacement text usually is the rest of the line, but long texts can be continued in multiple lines with \ at the end of the line
- $\bullet$  it is done for tokens quoted strings are not processed
- name can have parameters

ivotes			

NI - + - -



# Define – Example

#define MAX\_SIZE 100 #define MAX(A,B) ((A)>(B)?(A):(B)) #define FOREVER for(;;) /\*infinite loop\*/
#define dprint(expr) printf(#expr "==%g\n", expr)



## Conditional inclusion

Conditional inclusion provides us a way to control preprocessing and to include code selectively.

#if, #elif, #else and #endif

 $\mbox{\tt\#if}$  evaluates constant integer expression (except sizeof, cast and enum constants) if this expression is non-zero then subsequent lines are included until #elif, #else or #endif.

#### defined(name)

This expression has a value 1 if the name has been defined or  ${\bf 0}$ otherwise  $\# ifdef\ name\ and\ \# ifndef\ name\ can\ be\ used\ instead\ of$ #if defined(name) and #if !defined(name) respectively

Notes			



# Conditional inclusion – Example

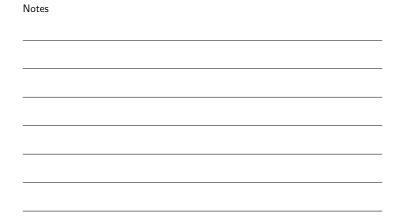
 ng	
C (2)	
slaw uk	
action	
øx	
ox	
ion	
ry ns	
n	
ns e	
ation	
TSSOP	



# What have we done today?

- 1 Introduction to Tests

  - Random tests
  - Black-box tests
  - Glass-box tests
  - Regression tests
  - Boundary conditions testing
  - Pre- and Post-condition testing
  - Assertions
  - Example
- 2 C-continuation
  - Functions
  - Scope
  - Preprocessor





## Next time

[KR] Chapter 5 Arrays Pointers

Notes			

Notes	
	_