CS35L Software Construction Laboratory

Lab 5: Sneha Shankar

Week 5; Lecture 2

Ternary Operator

- Short form for a conditional assignment

result = a > b ? x : y; is equivalent to:

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

Lab 4

- Download old version of coreutils with buggy ls program
  - Untar, configure, make
- Bug: ls -t mishandles files whose time stamps are very far in the past. It seems to act as if they are in the future

```
$ tmp=$(mktemp -d)
$ cd $tmp
$ touch -d '1918-11-11 11:00 GMT' wwi-armistice
$ touch now
$ sleep 1
$ touch now1
$ ls -lt wwi-armistice now now1
```

Output:

-rw-r--r-- 1 eggert eggert 0 Nov 11 1918 wwi-armistice

-rw-r--r-- 1 eggert eggert 0 Feb 5 15:57 now1

-rw-r--r-- 1 eggert eggert 0 Feb 5 15:57 now

Goal: Fix the Bug

- **Reproduce the Bug**
  - Follow steps on lab web page
- **Simplify input**
  - Run ls with –l and –t options only
- **Debug**
  - Use gdb to figure out what's wrong
  - $ gdb ./ls
  - (gdb) run –lt wwi-armistice now now1

  (run from the directory where the compiled ls lives)
- **Patch**
  - Construct a patch "lab5.diff" containing your fix
  - It should contain a ChangeLog entry followed by the output of diff -u

Lab Hints

- Use “info functions” to look for relevant starting point
- Use “info locals” to check values of local variables
- Compiler optimizations: -O2 -> -O0
  - ./configure CFLAGS="…-O0"

Task 1

Program Statement – Define a structure called student that will describe the following information.

- name (char *array)
- Uid (int)

Then create an array (of size 3) of this structure type.

struct student <array name>[3]; //access attributes using <array name>[index].attributename}

Using student, declare an array player with 3 elements and write a program to read the information about all the 3 players and print a sorted team wise list (sort by team name) containing names of students with their UIDs.

*you can hardcode the data for your convenience

Use the qsort function

Task 1 solution

```
int compare (const void * a, const void * b ) {
   struct student *pa = (struct student*)a;
  struct student *pb = (struct student*)b;
    return strcmp(pa->name, pb>name);
}
qsort(<arrayname>,5, sizeof(struct student),compare);
```

*you can also typedef to avoid writing ‘struct’

Initializing array using malloc

```
int *arr = malloc (sizeof (int) * n); /* n is the length of the array */
int i;

for (i=0; i<n; i++)
{
  arr[i] = 0;
}
```

Task 2

/*Using structures to calculate the area of a rectangle*/

Create two structs for Rectangle and Point.

Calculate the area of the rectangle using the given coordinates (top left and bottom right)

Use the below structure:

```
typedef struct {
    Point topLeft;   /* top left point of rectangle */
    Point botRight;  /* bottom right point of rectangle */
} Rectangle;
```

Task 2 Solution

```
#include <stdio.h>
#include <string.h>
#include <math.h>

typedef struct {
    double x;
    double y;
} Point;

typedef struct {
    Point topLeft;   /* top left point of rectangle */
    Point botRight;  /* bottom right point of rectangle */
} Rectangle;
double computeArea(Rectangle *r);
```

```
int main()
{
    Point p;
    Rectangle r;
    printf("\nEnter top left point: ");
    scanf("%lf", &r.topLeft.x);
    scanf("%lf", &r.topLeft.y);
    printf("Enter bottom right point: ");
    scanf("%lf", &r.botRight.x);
    scanf("%lf", &r.botRight.y);
    printf("Top left x = %lf  y = %lf\n", r.topLeft.x, r.topLeft.y);
    printf("Bottom right x = %lf  y = %lf\n", r.botRight.x, r.botRight.y);
    printf("Area = %f", computeArea(&r));
    return 0;
}
```

```
double computeArea(Rectangle *r)
{
    double height, width, area;

    height = ((r->topLeft.y) - (r->botRight.y));
    width = ((r->topLeft.x) - (r->botRight.x));
    area = height*width;
    return (area);
}
```

Task 3

Write a C program using getchar() and putchar() which continuously takes user input and prints it on the screen. This should keep on happening till the user inputs a string containing '#' and Enters.

Hint: use while(getchar() != #)

Task 3 solution

```
#include <stdio.h>
/* --  Copy input to output -- */
int main(void)
{
      int c;
      c = getchar();
      while ( c != "#" ) {
          putchar(c);
          c = getchar();
      }
      return 0;
}
```

Task 4

- Write the following line in a file called file.txt

  The value stored is 100
- Use fscanf to read the value 100 from file.txt and store it in a variable <var>.
- Then write this value to another file file1.txt “Value read is <var>” using fprintf

Task 4 solution

```
#include <stdio.h>
#include <stdlib.h>

int main() {
int a;
FILE * fp;
FILE * fp1;
fp = fopen("file.txt","r+");
fp1 = fopen("file1.txt", "w+");
fscanf(fp, "This is the value %d", &a);
fprintf(fp1, "Value read is %d",a);
fclose(fp);
return 0;
}
```

Gdb pointers

- Gdb cheat sheet
- Gdb command tutorial and slides
- Running gdb with emacs

Homework 4

- Write a C program called *sfrob*
  - Reads stdin byte-by-byte **(getchar)**
    - Consists of records that are newline-delimited
  - Each byte is frobnicated (XOR with dec 42)
    - Sort records without decoding (**qsort, frobcmp**)
    - Output result in frobnicated encoding to stdout **(putchar)**
  - Dynamic memory allocation (**malloc, realloc, free**)

Example

- Input: printf 'sybjre obl’
  - $ printf 'sybjre obl**\n**' | ./sfrob
- Read the records: sybjre, obl
- Compare records using *frobcmp* function
- Use *frobcmp* as compare function in *qsort*
- Output: obl

  sybjre

Homework Hints

- Array of pointers to char arrays to store strings (char ** arr)
- Use the right cast while passing frobcmp to qsort
  - cast from void * to char ** and then dereference because frobcmp takes a char *
- Use realloc to reallocate memory for every string and the array of strings itself, dynamically
- Use *exit*, not *return* when exiting with error