

CSC295 2014S, Class 12: Miscellaneous C Programming Issues

Overview

- Preliminaries.
 - Admin.
 - Questions.
- Memory issues in C programs.
- Checking memory problems with valgrind.
- Sorting, revisited.
- Function pointers.

Preliminaries

Admin

- I still need judges for the CSC 207 students Monday at 10:00 a.m.
- If you are going to be taking classes on campus this fall and you have not registered, do so ASAP!
- Congrats to AK!
- In honor of the Work/Life Balance Town Hall, there is no homework for next next week.
- Next week may be our last week of this class. We plan to have the CS senior celebration on the last Thursday of the semester.

Questions

Memory issues in C programs

- Great aspect of C: Memory is a first-class value.
- Also a huge problem; many program errors have to deal with failure to use memory correctly.
- How do programmers use memory incorrectly?
 - Allocated less than actually gets used. Uses more than allocated.
 - Typically allocated with malloc.
 - Typically used with arrays - outside bounds of arrays.
 - Allocating space for a data type, and use the wrong amount of space.

```
double *d = malloc(2);
```

- Run out of memory
- Attempt to write your own alternative to malloc and screw it up.
- Problems with freeing memory
 - You don't do it - Memory leak
 - You do it twice

```
int *ip = ...;
int *jp = ip;
...
free (ip);
free (jp);
```

- Memory issues sometimes have indirect effects

Checking memory problems with valgrind

- Catches a *lot* of different errors related to bad use of malloc (or allocation on the stack)

Sorting, revisited

Here's a standard signature

```
int strings_sort (int n, char *strings[]);
```

But wait! How do we specify what we want to sort by?

```
int strings_sort (int n, char *strings[], THING-TO-COMPARE-ELEMENTS compare);
```

Function pointers

```
int strings_sort (int n, char strings[], int (compare)(char *, char *))
```

How do we pass in a function that meets this signature?

- Just type the name of the function

```
strings_sort (10, names, strcmp);
```

```
int compare_by_length (char *str1, char *str2)
{
    if (strlen(str1) > strlen(str2))
        return 1;
    else if (strlen(str1) < strlen(str2))
```

```

        return -1;
    else
        return 0;
}

```

```
strings_sort (10, names, compare_by_length);
```

Making the type name a bit easier

```

typedef int (*StringComparator)(char *, char *);

int strings_sort (int n, char *strings[], StringComparator compare);

```

Within out implementation of strings_sort, how do we call compare?

Really precise version

```
(*compare)("alpha", "beta")
```

Normal C programmer version

```
compare("alpha", "beta")
```

Challenge! Write a program that calls strings_sort to sort the strings on the command line alphabetically

```

int
main (int argc, char *argv[])
{
    int i;
    strings_sort (argc-1, argv+1, strcmp);
    for (i = 0; i < argc-1; i++)
        printf ("%s ", argv[i]);
    printf ("\n");
} // main

```

Copyright (c) 2013-14 Samuel A. Rebelsky.



This work is licensed under a Creative Commons Attribution 3.0 Unported License. To view a copy of this license, visit <http://creativecommons.org/licenses/by/3.0/> or send a letter to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.