CS 486: Assignment 1
Library

Jon A. Solworth

10 Feb 2016

Due	**17 Feb 2016 at start of class**
Hand in	Code plus use document
Demo	Working code

1 Overview

C has so few built-in capabilities that the first job is to build some basic libraries that your OS (nanoOS) will use. Components of these libraries include memory/string manipulation, printing, basic data structures, and storage allocation.

These libraries are to be integrated into your nanoOS. This assignment requires that you integrate and test these libraries. You will create three files, `mem.h`, `mem.c`, and `test.c`.

2 Details

In this assignment you'll build the these routines:

```
void *memcpy(void *dest, const void *src, size_t n);
void *memset(void *s, int c, size_t n);
int memcmp(const void *s1, const void *s2, size_t n);
char *strcat(char *dest, const char *src);
char *strncat(char *dest, const char *src, size_t n);
int strcmp(const char *s1, const char *s2);
int strncmp(const char *s1, const char *s2, size_t n);
char *strcpy(char *dest, const char *src);
char *strncpy(char *dest, const char *src, size_t n);
int printf(const char *format, ...);
```

The meaning of the above APIs are the same as the user space routines, and are described in various man pages. You are expected to adhere to those manual pages for all routines except **printf**.

Printf is more complicated, and so we'll simplify it here. `Printf` should accept an arbitrary number of parameters in a single call. (You'll need to include `stdarg.h`.) After the format string, each argument is either an int, long, or a string. Your printf should support formats '%s', '%d', '%x', '%u', '%ld', '%lx', '%lu'. The result of printf is sent to stdout. The code should work on both 32-bit and 64-bit architectures.

3 Other issues

There is code for these routines on the Internet. If you look at any code, you must reference it in your code, describe how you have used it, and point to its source (URL).

You are not to show anyone else your code or to look at anyone else's code (we will be examining the code for similarity). Otherwise, you are free and encouraged to discuss the assignments.

You are expected to completely understand *any* code you hand in.

Hints

1. Build and test the code in user space,
2. Start immediately
3. You should look at as many issues in parallel (coding, testing, linking, etc.)
4. You may put a limit on printf output size or you may write it so there is no limit. You must state which is the case.
5. You must test all pointers to determine if they have a NULL value, and if so to end your program, after printing out an appropriate error message. You should use __FILE__ and __LINE__ to print an appropriate message.

4 What to hand in

- Test the different cases for your code
- Test for illegal issues such as more parameters than format specifiers, more format specifiers than parameters, parameter type mismatch, and interpret the results.
- In addition, I'd like to see the output of the following tests

```
char *cat = "cat";
char *dogGoesWoof = "dog\0goes\0woof"
char str[100];
memset(str,0,100);
strcat(str, cat);
printf("%s", str);
```

```
strcpy(str, dogGoesWoof);
printf("%s", str);
strcpy(str + 1, str);
memcpy(str, cat, 22);
char *ptr = 0;
*ptr = 'a';
```

(Note that these tests may do strange things, like crash your program. Thus, you may not be able to execute them all in a single run.)