CS 240 (Park) **Final** **Aug. 4 (Thu.), 2016**

Remarks: Keep the answers compact, yet precise and to-the-point. Long-winded answers that do not address the key points are of limited value. Binary answers that give little indication of understanding are no good either. Time is not meant to be plentiful. Make sure not to get bogged down on a single problem.

PROBLEM 1 (36 pts)

(a) The UNIX command *cp* takes two string arguments *file1* and *file2* (assume both represent ASCII files) and copies the content of *file1* byte-by-byte to *file2*. Write main() code that accomplishes that. For brevity, you may omit specifying header files and checking whether opening of the files succeeds or not.

(b) execlp() is a version of execl() that allows the programmer not to have to specify the full pathname of an executable. For example, simply *ls* in place of */bin/ls*. What must the code of execlp() do—assuming it is executed by a shell—to provide this feature? Explain in words, without coding, how execlp() internally works. All other parts of execlp() are essentially the same as execl() and, hence, can be ignored.

(c) What is the function prototype of the stdio fprintf() based on our discussion of how printf() works? In words, without writing code, explain what the source code of fprintf() must do. Suppose a programmer uses fprintf() to output five integers to a file. However, by mistake, the programmer lists a sixth integer variable when calling fprintf() even though the format string string only specifies five instances of "%d". What will happen, and why?

PROBLEM 2 (36 pts)

(a) The system call signal() allows a C program to register a callback function (i.e., signal handler) that the programmer wrote by passing as its second argument the address of the signal handler (i.e., function pointer). The first argument specifies an integer (e.g., SIGSEGV) that identifies which signal should cause the operating system (in our case Linux) to invoke the signal handler. Note that signal handlers are of type void and take a single integer argument which specifies the signal. Assuming signal() returns an integer, what is its function prototype? In the source code of the function signal(), how will it call the signal handler? Write a code snippet that does that.

(b) In the function prototype of execlp(), the first argument is defined as *const char *file*. What beneficial role does *const* serve? What issue may arise that impacts the programmer who calls execlp() if the first argument were defined as *char *file* without *const*? Can this issue be addressed without using *const*? Explain your reasoning.

(c) The second argument of main(), when used, is interchangeably declared as *char **argv* or *char *argv[]*. What do they mean and why are they equivalent? Suppose the copy command in Problem 1(a) is invoked at a shell prompt as follows: *% cp input.dat output.dat*. How is *argv* in main() of the app *cp* represented in memory (i.e., RAM)? Sketch a diagram of an example memory layout as we have done in class for the above command.

PROBLEM 3 (28 pts)

The following data structure

*typedef struct mystuff {char *name; int value; mystuff_t *next} mystuff_t;*

may be used to remember all the stuff you own and what you paid for them for insurance purposes. The first field specifies the name of an item, the second field its dollar value, and the third contains the address of memory where the next item you purchased is remembered. The last item has the next field set to NULL to indicate that it's last. Two variables are declared as *mystuff_t *xstart, *xend;* where xstart points to the first item you purchased and xend points to the last. Initially you own nothing, hence both xstart and xend are set to NULL. Suppose you buy a new item (e.g., F250 50000) whose name and value you enter through standard input to keep track of things. Write an app implemented as a single function main() that reads the new item's name and value from stdin, uses malloc() to allocate memory for it, then links it up with the previously last item pointed to by xend. After linking up, xend should point to the item you just bought. When the item you just bought is your first, then xstart and xend point to the same memory location.

BONUS PROBLEM (10 pts)

Using a function with a local string variable, char a[100], explain the meaning of stack smashing. What does gcc do to help detect stack smashing?