**Student**: Michael Escue

**Assignment**: Assignment #1

**Class**: ECE 373

**Instructors**: Peter (PJ) Waskiewicz,

Charles Stoll

**Term**: Spring 2019

**A. Information Gathering:**

**1. Look at the man page for the "open()" system call, found in section two of the manpages.**

**A. List the BUGS when using the system call.**

﻿

|  |
| --- |
| * Currently, it is not possible to enable signal-driven I/O by specifying   O\_ASYNC when calling open(); use fcntl(2) to enable this flag.   * One must check for two different error codes, EISDIR and ENOENT, when   trying to determine whether the kernel supports O\_TMPFILE functional‐  ity.   * When both O\_CREAT and O\_DIRECTORY are specified in flags and the file   specified by pathname does not exist, open() will create a regular file  (i.e., O\_DIRECTORY is ignored). |

**B. What files need to be included to use this function?**

|  |
| --- |
| * #include <sys/types.h> * #include <sys/stat.h> * #include <fcntl.h> |

**C. List the first three related system calls to open().**

|  |
| --- |
| * chmod(2), chown(2), close(2) |

**D. Choose one of the system calls from above and list its bugs (also list what system call you chose) and files needing to be included to use the system call.**

|  |
| --- |
| * Chosen sys call: Close( ) * #include <unistd.h> * No list of bugs in the man page. Wasn’t able to directly find any bugs using the link included for reporting bugs. There are a list of 44 unresolved bugs, but no titles.   “information about reporting bugs, and the latest version of this page, found at <https://www.kernel.org/doc/man-pages/>.”   * Bug list:   <https://bugzilla.kernel.org/buglist.cgi?bug_status=NEW&bug_status=ASSIGNED&bug_status=REOPENED&email1=documentation_man-pages%40kernel-bugs.osdl.org&emailassigned_to1=1&emailtype1=substring&order=component%20DESC%2Cbug_status%2Cpriority%2Cassigned_to%2Cbug_id&query_format=advanced> |

**2. Use http://lxr.free-electrons.com to search for the following:**

**A. Search for "usb\_device". In what file is the structure defined, and what are the first five members of the struct?**

|  |
| --- |
| * Version: Linux v5.0.7 * include/linux/usb.h * First five members:   int devnum;  char devpath[16];  u32 route;  enum [usb\_device\_state](https://elixir.bootlin.com/linux/latest/ident/usb_device_state) state;  enum [usb\_device\_speed](https://elixir.bootlin.com/linux/latest/ident/usb_device_speed) [speed](https://elixir.bootlin.com/linux/latest/ident/speed); |

**B. In what header file is the type declared for the 5th member of the struct? (hint: don't look in test tools or staging directories)**

|  |
| --- |
| * [include/uapi/linux/usb/ch9.h](https://elixir.bootlin.com/linux/latest/source/include/uapi/linux/usb/ch9.h#L1137) |

**C. Include the entire enumeration declaration from above.**

|  |
| --- |
| * Enumeration in header:   enum [usb\_device\_speed](https://elixir.bootlin.com/linux/latest/ident/usb_device_speed) {  [USB\_SPEED\_UNKNOWN](https://elixir.bootlin.com/linux/latest/ident/USB_SPEED_UNKNOWN) = 0, /\* enumerating \*/  [USB\_SPEED\_LOW](https://elixir.bootlin.com/linux/latest/ident/USB_SPEED_LOW), [USB\_SPEED\_FULL](https://elixir.bootlin.com/linux/latest/ident/USB_SPEED_FULL), /\* usb 1.1 \*/  [USB\_SPEED\_HIGH](https://elixir.bootlin.com/linux/latest/ident/USB_SPEED_HIGH), /\* usb 2.0 \*/  [USB\_SPEED\_WIRELESS](https://elixir.bootlin.com/linux/latest/ident/USB_SPEED_WIRELESS), /\* wireless (usb 2.5) \*/  [USB\_SPEED\_SUPER](https://elixir.bootlin.com/linux/latest/ident/USB_SPEED_SUPER), /\* usb 3.0 \*/  [USB\_SPEED\_SUPER\_PLUS](https://elixir.bootlin.com/linux/latest/ident/USB_SPEED_SUPER_PLUS), /\* usb 3.1 \*/  }; |

**B. Basic Linux Use:**

**1. Log onto an Ubuntu Linux system**

**2. Under each heading in the toolbar at the top, pop up and note the menu and sub menu tools/topics. Note where you find Terminal and the gedit Text Editor.**

**3. Click on the Terminal entry to bring up a command shell line. This should be familiar to you from working with Cygwin in ECE 371/372. You will use Terminal to enter Linux commands.**

**4. To get an overview of Linux structure and commands, go to http://gd.tuwien.ac.at/linuxcommand.org/learning\_the\_shell.php . Study the “Learning the shell” section carefully and make some notes on basic commands to make a directory, determine current directory, change to a different directory, do a long or short form listing of the files in a directory, copy a file, etc.**

**5. The “script” command is a useful way to make a record of all the commands you enter at the command shell and the results of those commands. Script records the commands, etc. in a file called typescript in your current directory. You terminate script with CTRL-D or 'exit'. You can then edit/print the typescript file. You will be turning in a printout of the typescript file for a shell practice session.**

**6. For a start with the basic shell commands, enter script at the command prompt and then:**

**A. Enter the command to determine the current directory**

**B. Enter the command to command to show the files in that directory with file permissions.**

**C. Enter a command to make a directory called ECE373.**

**D. Change the working directory to that directory.**

**E. Exit script with CTRL D.**

**7. Find the gedit program under Application|Accessories and open a gedit window. Find the File|Open menu command and find the typescript file you created. Click on Open and you should see the typescript file in the window. It likely contains some control characters along with the commands but you can easily edit these out, if you want. Click on the Printer icon in gedit to print the typescript file.**

|  |
| --- |
| * **michael@michael-VirtualBox:~/Documents/ECE373/Assignments/HW1$** cat typescript   Script started on 2019-04-13 22:55:01-0700  michael@michael-VirtualBox: ~/Documents$ pwd  /home/michael/Documents  michael@michael-VirtualBox: ~/Documents$ ls -l  total 0  -rw-r--r-- 1 michael michael 0 Apr 13 22:55 typescript  michael@michael-VirtualBox: ~/Documents$ mkdir -v ECE373  mkdir: created directory 'ECE373'  michael@michael-VirtualBox: ~/Documents$ cd ECE373  michael@michael-VirtualBox: ~/Documents/ECE373$ exit  Script done on 2019-04-13 22:56:38-0700 |

**C. Basic C Programming in Linux:**

***The tools you will use for developing C programs for this class are the GNU tools, gcc, ln, and gdb. For a start with these, work through the following steps.***

**1. Type in a simple C program such as a Fahrenheit to Celsius program or one of the examples from the C reference that takes in data and outputs a result to the display.**

|  |
| --- |
| /\*\*  \* @file hw1.c  \* @author Michael Escue  \* @brief Fahrenheit to Clesius conversion program.  \* @version 0.1  \* @date 2019-04-14  \*  \* @copyright Copyright (c) 2019  \*  \*/  #include "hw1.h"  /\*\*  \* @brief Defines for program.  \*  \*/  #define N 256  #define A 32  /\*\*  \* @brief inline code for prompt, scan, display conversion results.  \*  \* @return int  \*/  int main(void){  float temp = 5.0/9.0;  float F = 0.0;  float C = 0.0;  char buf[N] = {0};  float \*Fp = &F;  printf("Enter degrees in Fahrenheit \"ff.f\": ");  scanf("%f", Fp);  getchar();  printf("\n");  //Operations  temp = ((F-A)\*temp);  printf("Result: %f degrees C\n", temp);    return 1;  } |

**2. To compile your C program, enter the command gcc -g -o filename filename.c where filename is the name you gave your C file. The –g switch tells the compiler to include debug information that will be used by the gdb debugger.**

**3. Edit and recompile the .c file until you get no compile errors.**

**4. To run the program from the command line, just enter ./filename.**

**5. If the program functions correctly, rejoice, turn off script, print the typescript file, and print the C source file.**

|  |
| --- |
| * Script started on 2019-04-14 00:27:23-0700   michael@michael-VirtualBox:~/Documents/ECE373$ gcc hw1.c -g -o c-to-f  hw1.c: In function ‘main’:  hw1.c:19:5: warning: implicit declaration of function ‘printf’ [-Wimplicit-function-declaration]  printf("Enter degrees in Fahrenheit \"ff.f\": ");  ^~~~~~  hw1.c:19:5: warning: incompatible implicit declaration of built-in function ‘printf’  hw1.c:19:5: note: include ‘<stdio.h>’ or provide a declaration of ‘printf’  hw1.c:20:5: warning: implicit declaration of function ‘scanf’ [-Wimplicit-function-declaration]  scanf("%f", Fp);  ^~~~~  hw1.c:20:5: warning: incompatible implicit declaration of built-in function ‘scanf’  hw1.c:20:5: note: include ‘<stdio.h>’ or provide a declaration of ‘scanf’  hw1.c:21:5: warning: implicit declaration of function ‘getchar’ [-Wimplicit-function-declaration]  getchar();  ^~~~~~~  michael@michael-VirtualBox:~/Documents/ECE373$ ls -l  total 20  -rwxrwxr-x 1 michael michael 9744 Apr 14 00:27 c-to-f  -rw-r--r-- 1 michael michael 437 Apr 13 23:45 hw1.c  -rw-r--r-- 1 michael michael 104 Apr 13 23:28 hw1.h  -rw-rw-r-- 1 michael michael 0 Apr 14 00:27 typescript  michael@michael-VirtualBox:~/Documents/ECE373$ ./c-to-f  Enter degrees in Fahrenheit "ff.f": 32.0  Result: 0.000000  michael@michael-VirtualBox:~/Documents/ECE373$ ./c-to-f  Enter degrees in Fahrenheit "ff.f": 0.0  Result: -17.777779  michael@michael-VirtualBox:~/Documents/ECE373$ ./c-to-f  Enter degrees in Fahrenheit "ff.f": 60.5  Result: 15.833334  michael@michael-VirtualBox:~/Documents/ECE373$ exit  Script done on 2019-04-14 00:30:51-0700 |

**6. If the program does not function correctly or even if it does, it is time to learn about gdb.**

**7. At the command prompt, enter man gdb. This should bring up the manual pages for gdb.**

**8. Read through the pages using the arrow keys to find out how to use gdb. Write some notes about the main gdb commands such as break, run, next, step, print, and quit. Hit the q key to exit the man pages.**

**9. Try running your program under gdb with a gdb filename command. Enter a breakpoint at main and then run the program. Execution then stops at main(). You can step through and execute one line of the program with step or next (note the difference). You can determine the values of variables at any point with the print command.**

**10. Experiment with gdb until you are comfortable working through a program with it.**

**D. Hello, Kernel:**

***Now that you have the basics, we get to the real point of the whole assignment: load a simple “hello” module into the kernel.***

**1. Following the notes from the lecture, create a simple hello.c file with the basic \_\_init and \_\_exit functions, where the init function prints “Hello, Kernel” and the exit function prints “Goodbye, Kernel”.**

|  |
| --- |
| /\*\*  \* @file hello.c  \* @author Michael Escue  \* @brief A file for implementing hello module.  \* @version 0.1  \* @date 2019-04-14  \*  \* @Code provided by Peter (PJ) Waskiewicz  \*  \*/  #include <linux/init.h>  #include <linux/module.h>  MODULE\_LICENSE("Dual BSD/GPL");  static int \_\_init hello\_init(void){  printk(KERN\_INFO "Hello, kernel\n");  return 0;  }  static void \_\_exit hello\_exit(void){  printk(KERN\_INFO "Goodbye, kernel\n");  }  module\_init(hello\_init);  module\_exit(hello\_exit); |

**2. Create a simple makefile for compiling the hello.c file into a kernel module**

|  |
| --- |
| # @file hw1.c Makefile  # @author Michael Escue  # @brief Makefile for HW1  # @version 0.1  # @date 2019-04-14  # @copyright Copyright (c) 2019    C = gcc  HW1\_OBJS = hw1.o  HELLO\_SRC = hello.c  HW1\_SRC = hw1.c  HW1\_HDRS = hw1.h  HELLO\_OBJS = hello.o  # From Lecture  KERNEL\_DIR = /lib/modules/$(shell uname -r)/build  PWD := $(shell pwd)  obj-m += hello\_kernel.o  # Make file code:  #all: hw1 default  default:  $(MAKE) -C $(KERNEL\_DIR) SUBDIRS=$(PWD) modules  #hw1: hw1.o  # $(C) -o $@ $(HW1\_OBJS)  #hw1.0: hw1.c $(HW1\_HDRS)  #$(C) -c $(HW1\_SRC)    clean:  #rm $(HW1\_OBJS)  # rm $(HELLO\_OBJS)  # rm hello\_kernel  # rm hw1  $(MAKE) -C $(KERNEL\_DIR) SUBDIRS=$(PWD) clean |

**3. Compile the hello.c file into hello.ko, re-editing as necessary to clean up any and all compiler warnings and errors.**

**4. Load the module into the kernel with the insmod utility and look for the “Hello, Kernel” message in the log file /var/log/messages**

**5. Remove the module from the kernel with the rmmod utility and look for the “Goodbye, Kernel” message in the log file.**

|  |
| --- |
| **michael@michael-VirtualBox:~/Documents/ECE373/Assignments/HW1$** make  make -C /lib/modules/4.18.0-15-generic/build SUBDIRS=/home/michael/Documents/ECE373/Assignments/HW1 modules  make[1]: Entering directory '/usr/src/linux-headers-4.18.0-15-generic'  Building modules, stage 2.  MODPOST 1 modules  make[1]: Leaving directory '/usr/src/linux-headers-4.18.0-15-generic'  #gcc -c hw1.c  lo\_kernel.koael-VirtualBox:~/Documents/ECE373/Assignments/HW1$ sudo insmod hel  [sudo] password for michael:  o\_kernel.kohael-VirtualBox:~/Documents/ECE373/Assignments/HW1$ sudo rmmod hell  **michael@michael-VirtualBox:~/Documents/ECE373/Assignments/HW1$** dmesg  …  [19257.432304] Hello, kernel  [19268.111347] Goodbye, kernel  **michael@michael-VirtualBox:~/Documents/ECE373/Assignments/HW1$** exit |

**What to turn in:**

In a text file, or Word document, or Open Office document, include:

1) For the manpage exercise, all of the information from #1 in "The Work" above.

2) For the LXR exercise, copy and paste the lines of code the LXR hits on PLUS the kernel you

searched against. For example, searching for sys\_gettimeofday() on kernel 3.14, then the line of

code would be lines 203 and 204, and the code would be:

asmlinkage long sys\_gettimeofday(struct timeval \_\_user \*tv,

struct timezone \_\_user \*tz);

Also include the file names requested in each of the parts of the exercise above.

3) Print of the example C program from section C.

4) Print of the typescript showing execution of program from section C.

5) Print of the hello.c module code from section D.

6) Print of the 'hello' and 'goodbye' logfile lines from section D.

Save all this information in the aforementioned file, print it, and turn it in at the **START of class on Monday, 15-April-2019**.