ECE 373 - Embedded Operating Systems & Device Drivers **Spring 24**
Assignment #1 Wail (Wa'el) AL KALBANI

Part A: Information Gathering

1.

A.

- 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 functionality.
- 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.

- Need to include <fcntl.h> to use the open() function.

C.

- The first three related system calls might include creat(), close(), and fcntl().

D.

- Chosen system call: fcntl()
- Bugs:
    - F_SETFL: It is not possible to use `F_SETFL` to change the state of the `O_DSYNC` and `O_SYNC` flags. Attempts to change the state of these flags are silently ignored.
    - F_GETOWN: On some architectures (notably i386), a limitation of the Linux system call conventions means that if a (negative) process group ID to be returned by `F_GETOWN` falls in the range -1 to -4095, then the return value is wrongly interpreted by glibc as an error in the system call. The Linux-specific `F_GETOWN_EX` operation avoids this problem.
    - F_SETOWN: In Linux 2.4 and earlier, there is a bug that can occur when an unprivileged process uses `F_SETOWN` to specify the owner of a socket file descriptor as a process (group) other than the caller. Despite an error return, the file descriptor owner is set, and signals will be sent to the owner.
    - Deadlock detection: The deadlock-detection algorithm employed by the kernel when dealing with `F_SETLKW` requests can yield both false negatives (failures to detect deadlocks) and false positives (EDEADLK errors when there is no deadlock).
    - Mandatory locking: The Linux implementation of mandatory locking is subject to race conditions which render it unreliable. It is therefore inadvisable to rely on mandatory locking.
    - To use the `fcntl()` function in code:
      #include <fcntl.h>   // include the `<fcntl.h>` header file

2.

A.

- Defined in usb.h file (/nclude/linux/usb.h).
- The first five members of the struct are:
  (1) int devnum;
  (2) char devpath[16];
  (3) u32 route;
  (4) enum usb_device_state state;
  (5) enum usb_device_speed speed;

B.

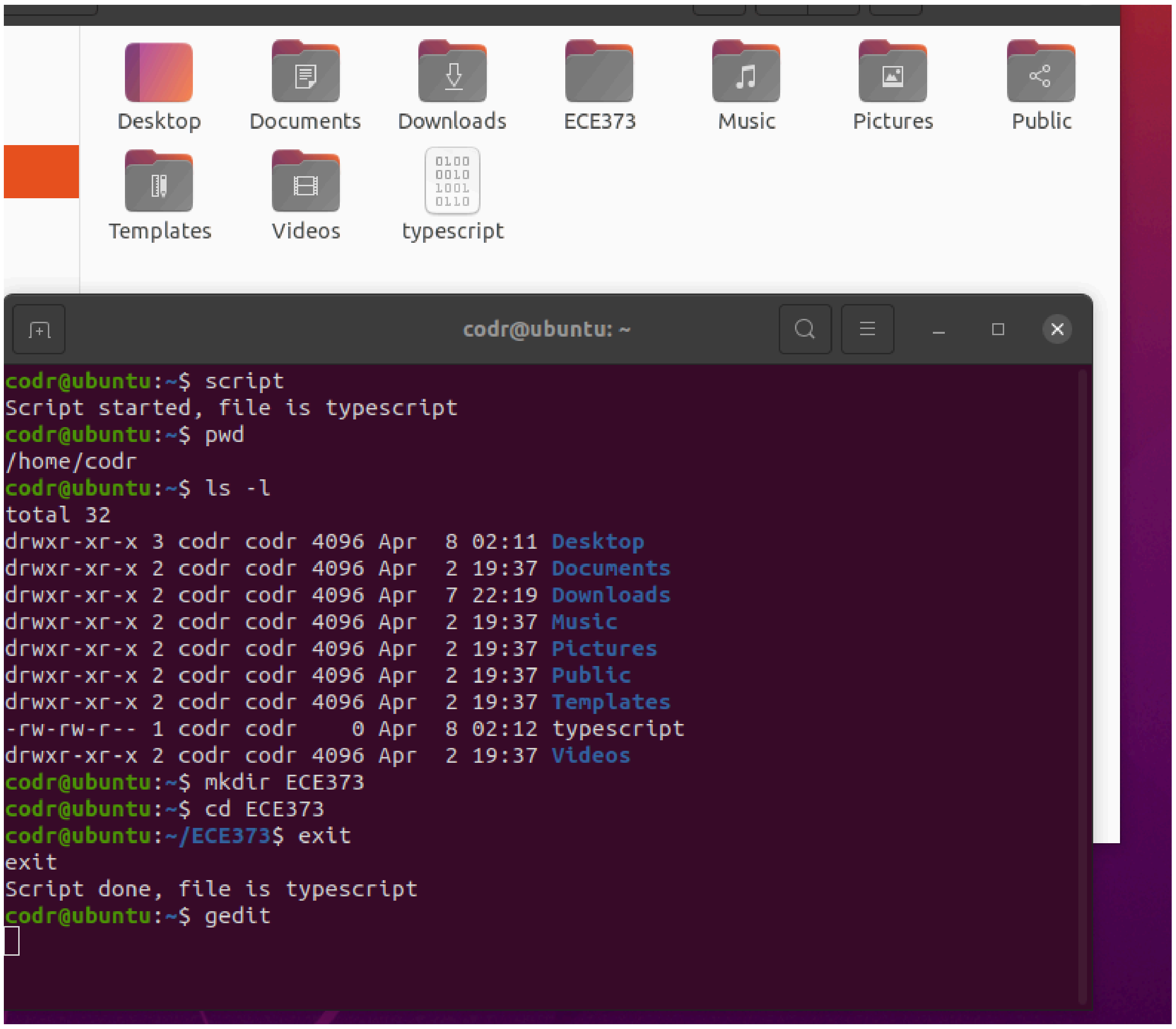
- Declared in ch9.h (/include/uapi/linux/usb/ch9.h).

C.

```
/* USB 2.0 defines three speeds, here's how Linux identifies them */

enum usb_device_speed {
        USB_SPEED_UNKNOWN = 0,                    /* enumerating */
        USB_SPEED_LOW, USB_SPEED_FULL,            /* usb 1.1 */
        USB_SPEED_HIGH,                   /* usb 2.0 */
        USB_SPEED_WIRELESS,                       /* wireless (usb 2.5) */
        USB_SPEED_SUPER,                  /* usb 3.0 */
        USB_SPEED_SUPER_PLUS,                     /* usb 3.1 */
};
```

Part B: Basic Linux Use



```
codr@ubuntu:~$ script
Script started, file is typescript
codr@ubuntu:~$ pwd
/home/codr
codr@ubuntu:~$ ls -l
total 32
drwxr-xr-x 3 codr codr 4096 Apr  8 02:11 Desktop
drwxr-xr-x 2 codr codr 4096 Apr  2 19:37 Documents
drwxr-xr-x 2 codr codr 4096 Apr  7 22:19 Downloads
drwxr-xr-x 2 codr codr 4096 Apr  2 19:37 Music
drwxr-xr-x 2 codr codr 4096 Apr  2 19:37 Pictures
drwxr-xr-x 2 codr codr 4096 Apr  2 19:37 Public
drwxr-xr-x 2 codr codr 4096 Apr  2 19:37 Templates
-rw-rw-r-- 1 codr codr    0 Apr  8 02:12 typescript
drwxr-xr-x 2 codr codr 4096 Apr  2 19:37 Videos
codr@ubuntu:~$ mkdir ECE373
codr@ubuntu:~$ cd ECE373
codr@ubuntu:~/ECE373$ exit
exit
Script done, file is typescript
codr@ubuntu:~$ gedit
```

Part C:Basic C Programming in Linux

Program - GNU edited example

hello.c

```
/* hello.c -- print a greeting message and exit.

   Copyright (C) 1992, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002,
   2005, 2006, 2007 Free Software Foundation, Inc.

   This program is free software; you can redistribute it and/or modify
   it under the terms of the GNU General Public License as published by
   the Free Software Foundation; either version 3, or (at your option)
   any later version.
   This program is distributed in the hope that it will be useful,
   but WITHOUT ANY WARRANTY; without even the implied warranty of
   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
   GNU General Public License for more details.

   You should have received a copy of the GNU General Public License
   along with this program; if not, write to the Free Software Foundation,
   Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.

   * Edited by: Wa'el AL KALBANI
   * Course: ECE 373 - Embedded Operating Systems & Device Drivers
 *
*/

#include "config.h"
#include "system.h"

/* String containing name the program is called with.  */
const char *program_name;

static const struct option longopts[] =
{
  { "greeting", required_argument, NULL, 'g' },
  { "help", no_argument, NULL, 'h' },
  { "next-generation", no_argument, NULL, 'n' },
  { "traditional", no_argument, NULL, 't' },
  { "version", no_argument, NULL, 'v' },
  { NULL, 0, NULL, 0 }
};

static void print_help (void);
static void print_version (void);

int
main (int argc, char *argv[])
{
  int optc;
  int t = 0, n = 0, lose = 0;
  const char *greeting = NULL;

  program_name = argv[0];

  /* Set locale via LC_ALL.  */
  setlocale (LC_ALL, "");

#if ENABLE_NLS
  /* Set the text message domain.  */
  bindtextdomain (PACKAGE, LOCALEDIR);
  textdomain (PACKAGE);
#endif

  /* Even exiting has subtleties.  The /dev/full device on GNU/Linux
     can be used for testing whether writes are checked properly.  For
     instance, hello >/dev/full should exit unsuccessfully.  On exit,
     if any writes failed, change the exit status.  This is
     implemented in the Gnulib module "closeout".  */
  atexit (close_stdout);

  while ((optc = getopt_long (argc, argv, "g:hntv", longopts, NULL)) != -1)
    switch (optc)
      {
      /* One goal here is having --help and --version exit immediately,
         per GNU coding standards.  */
      case 'v':
        print_version ();
        exit (EXIT_SUCCESS);
        break;
      case 'g':
        greeting = optarg;
        break;
      case 'h':
        print_help ();
        exit (EXIT_SUCCESS);
        break;
      case 'n':
        n = 1;
        break;
      case 't':
        t = 1;
        break;
      default:
        lose = 1;
        break;
      }

  if (lose || optind < argc)
    {
      /* Print error message and exit.  */
      if (optind < argc)
        fprintf (stderr, _("%s: extra operand: %s\n"),
         program_name, argv[optind]);
      fprintf (stderr, _("Try `%s --help' for more information.\n"),
               program_name);
      exit (EXIT_FAILURE);
    }

  /* Print greeting message and exit. */
  if (t)
    printf (_("hello, world\n"));

  else if (n)
    /* TRANSLATORS: Use box drawing characters or other fancy stuff
       if your encoding (e.g., UTF-8) allows it.  If done so add the
       following note, please:

       [Note: For best viewing results use a UTF-8 locale, please.]
    */
    printf (_("\
+---------------+\n\
| Hello, world! |\n\
+---------------+\n\
"));

  else
    {
      if (!greeting)
        greeting = _("Hello, world!");
      puts (greeting);
    }

  exit (EXIT_SUCCESS);
}



/* Print help info.  This long message is split into
   several pieces to help translators be able to align different
   blocks and identify the various pieces.  */

static void
print_help (void)
{
  /* TRANSLATORS: --help output 1 (synopsis)
     no-wrap */
        printf (_("\
Usage: %s [OPTION]...\n"), program_name);

  /* TRANSLATORS: --help output 2 (brief description)
     no-wrap */
  fputs (_("\
Print a friendly, customizable greeting.\n"), stdout);

  puts ("");
  /* TRANSLATORS: --help output 3: options 1/2
     no-wrap */
  fputs (_("\
  -h, --help          display this help and exit\n\
  -v, --version       display version information and exit\n"), stdout);

  puts ("");
  /* TRANSLATORS: --help output 4: options 2/2
     no-wrap */
  fputs (_("\
  -t, --traditional       use traditional greeting format\n\
  -n, --next-generation   use next-generation greeting format\n\
  -g, --greeting=TEXT     use TEXT as the greeting message\n"), stdout);

  printf ("\n");
  /* TRANSLATORS: --help output 5 (end)
     TRANSLATORS: the placeholder indicates the bug-reporting address
     for this application.  Please add _another line_ with the
     address for translation bugs.
     no-wrap */
  printf (_("\
Report bugs to <%s>.\n"), PACKAGE_BUGREPORT);
}
```

```
/* Print version and copyright information.  */

static void
print_version (void)
{
  printf ("hello (GNU %s) %s\n", PACKAGE, VERSION);
  /* xgettext: no-wrap */
  puts ("");

  /* It is important to separate the year from the rest of the message,
     as done here, to avoid having to retranslate the message when a new
     year comes around.  */
  printf (_("\
Copyright (C) %s Free Software Foundation, Inc.\n\
License GPLv3+: GNU GPL version 3 or later\
<http://gnu.org/licenses/gpl.html>\n\
This is free software: you are free to change and redistribute it.\n\
There is NO WARRANTY, to the extent permitted by law.\n"),
              "2007");
}
```

typescript

```
codr@ubuntu:~/ECE373/assignment1/c_gnu$ ls
closeout.h  config.h  gettext.h  hello.c  system.h
codr@ubuntu:~/ECE373/assignment1/c_gnu$ gcc -o hello hello.c
codr@ubuntu:~/ECE373/assignment1/c_gnu$ gdb hello
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04.1) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from hello...
(No debugging symbols found in hello)
(gdb) break main
Breakpoint 1 at 0x12b4
(gdb) step
The program is not being run.
(gdb) step
The program is not being run.
(gdb) run
Starting program: /home/codr/ECE373/assignment1/c_gnu/hello

Breakpoint 1, 0x00005555555552b4 in main ()
```

```
(gdb) step
Single stepping until exit from function main,
which has no line number information.
__GI_setlocale (category=6, locale=0x55555555603a "") at setlocale.c:218
218 setlocale.c: No such file or directory.
(gdb) step
225 in setlocale.c
(gdb) step
230 in setlocale.c
(gdb) step
234 in setlocale.c
(gdb) step
236 in setlocale.c
(gdb) step
217 in setlocale.c
(gdb) next
0x0000555555555306 in main ()
(gdb) next
Single stepping until exit from function main,
which has no line number information.
Hello, world!
[Inferior 1 (process 12223) exited normally]
(gdb) next
The program is not being run.
(gdb) continue
The program is not being run.
(gdb) run
Starting program: /home/codr/ECE373/assignment1/c_gnu/hello

Breakpoint 1, 0x00005555555552b4 in main ()
(gdb) continue
Continuing.
Hello, world!
[Inferior 1 (process 12228) exited normally]
(gdb) quit
codr@ubuntu:~/ECE373/assignment1/c_gnu$ ./hello
Hello, world!
```

Part D:Hello Kernel

hello.c

```
/*
 *
 * This is a simple kernel module for the ECE373 course.
 * It demonstrates the basics of creating a kernel module,
 * including initialization and cleanup functions.
 * When loaded, it prints "Hello, Kernel" to the kernel log,
 * and when unloaded, it prints "Goodbye, Kernel".
 *
 * Author: Wa'el AL KALBANI
 * Course: ECE 373 - Embedded Operating Systems & Device Drivers
 *
 */


#include <linux/module.h>   // "module.h" Needed for all kernel modules
#include <linux/version.h>  // "version.h" Needed for version checking
#include <linux/kernel.h>   // "kernal.h" Needed for kernel functions



// Initialization function
// __init to indicate that it is only used at initialization time.
static int __init init_HelloLinux(void){
    printk(KERN_INFO "Hello, Kernel \n");  // Print "Hello, Kernel" to the log
    return 0;                              // Return 0 to indicate successful
initialization
}


// Cleanup function
// __exit to indicate that it is only used at cleanup time.
static void __exit exit_HelloLinux(void){
    printk(KERN_INFO "Goodbye, Kernel \n");  // Print "Goodbye, Kernel" to the
log
}



module_init(init_HelloLinux);  // Register initialization function
module_exit(exit_HelloLinux);  // Register cleanup function


MODULE_LICENSE("GPL");  // License "GPL" which stands for GNU General Public
License
MODULE_AUTHOR("Wa'el AL KALBANI");  // Author declaration
MODULE_DESCRIPTION("ECE373 1st Assignment");  // Module description
```

logfile

```
codr@ubuntu:~$ cd ece373/assignment1
codr@ubuntu:~/ece373/assignment1$ ls
hello.c  Makefile
codr@ubuntu:~/ece373/assignment1$ make
make -C /lib/modules/5.15.0-101-generic/build M=/home/codr/ece373/assignment1
modules
make[1]: Entering directory '/usr/src/linux-headers-5.15.0-101-generic'
  CC [M]  /home/codr/ece373/assignment1/hello.o
  MODPOST /home/codr/ece373/assignment1/Module.symvers
  CC [M]  /home/codr/ece373/assignment1/hello.mod.o
  LD [M]  /home/codr/ece373/assignment1/hello.ko
  BTF [M] /home/codr/ece373/assignment1/hello.ko
Skipping BTF generation for /home/codr/ece373/assignment1/hello.ko due to
unavailability of vmlinux
make[1]: Leaving directory '/usr/src/linux-headers-5.15.0-101-generic'
codr@ubuntu:~/ece373/assignment1$ sudo lsmod
codr@ubuntu:~/ece373/assignment1$ sudo lsmod|grep hello
```

```
codr@ubuntu:~/ece373/assignment1$ sudo insmod hello.ko
codr@ubuntu:~/ece373/assignment1$ ls
hello.c   hello.mod    hello.mod.o  Makefile       Module.symvers
hello.ko  hello.mod.c  hello.o      modules.order
codr@ubuntu:~/ECE373/assignment1/d_kernel$ sudo dmesg | tail
[20069.050214] [  12329]  1000 12329    15475      349   118784        0
100 firefox
[20069.050216] [  12330]  1000 12330  1055342    40860  1789952    19999
0 Chroot Helper
[20069.050218] [  12333]  1000 12333     7691      158    65536        0
100 firefox
[20069.050220] [  12334]  1000 12334  1059698    41067  1806336    19995
0 Chroot Helper
[20069.050221]
oom-kill:constraint=CONSTRAINT_NONE,nodemask=(null),cpuset=/,mems_allowed=0,g
lobal_oom,task_memcg=/user.slice/user-1000.slice/user@1000.service,task=Isola
ted Web Co,pid=7535,uid=1000
[20069.050254] Out of memory: Killed process 7535 (Isolated Web Co)
total-vm:3346984kB, anon-rss:586908kB, file-rss:0kB, shmem-rss:13824kB,
UID:1000 pgtables:3780kB oom_score_adj:100
[20291.969240] hello: loading out-of-tree module taints kernel.
[20291.969276] hello: module verification failed: signature and/or required
key missing - tainting kernel
[20291.969442] Hello, Kernel
[20304.147863] Goodbye, Kernel
```