SANIKA DONGRE

Advanced Embedded Software Development

Homework 2 (100 Pts) - Due Sunday February 10th (before midnight)

Revision 2019.01.25

1 Guidelines

For this homework, both Canvas and Github online materials need to be submitted. **Turn in a *.pdf for your report to this assignment to Canvas. Please neatly format submissions with your name, date, homework # at the top, and enumerator your answers using headings indicating each problem number.**

And please format your answers - provide headings to the problems, context with captured data and descriptions for images and screenshots.

2 Reading & Resources

These are reading assignments that are good to complete the homework for the week as well as review materials to prepare for the upcoming content in the class.

- (MELP) Mastering Embedded Linux Programming (Second Edition) – Simmonds
  - Chapter 6 – Focus is on Buildroot and Adding your Own Code (Overlay)
- Linux Kernel Development – Love
  - Chapter 5 - System Calls (all)
  - Chapter 17 – Devices and Modules
- Linux Device Drivers – Corbet (skim/reference) https://lwn.net/Kernel/LDD3/
  - Chapters 1 & 2

Resources:

These are useful documents that will assist your work with the Buildroot toolchain and is an accompaniment to the MELP book, above.

- Buildroot
  - Buildroot Practical Labs - https://bootlin.com/doc/training/buildroot/buildroot-labs.pdf
  - Buildroot Manual - https://buildroot.org/downloads/manual/manual.pdf

These sections are not required but may help with doing the homework assignments.

- MELP - Chapter 3 – All about Bootloaders
- Ubuntu Linux Boot Procedure: https://wiki.ubuntu.com/Booting

3 Problem Set

[Problem 1] Record your Repository

Provide us a Github link to your repositories. Include a link to the repository on the top of your assignment to turn in.

SOLUTION:
https://github.com/sanikadongre/Advanced_Embedded_Software_Development

[Problem 2 - 20 Pts] Track system calls and library calls with File IO

Let' refresh your file operations skills, as this will be important as we move towards implementing a device driver and writing programs that require user interaction. Additionally, you'll familiarize yourself with and use tools to support your program's development. Specifically, you'll use perf, ltrace and strace to collect information on your program that reads and writes to a file on your development host machine. Your goal is to write a program that can:

- Print to standard out an interesting string using printf
- Create a file
- Modify the permissions of the file to be read/write
- Open the file (for writing)
- Write a character to the file
- Close the file
- Open the file (in append mode)
- Dynamically allocate an array of memory
- Read an input string from the command line and write to the string to the allocated array
- Write the string to the file
- Flush file output
- Close the file
- Open the file (for reading)
- Read a single character (getc)
- Read a string of characters (gets)
- Close the file
- Free the memory

After writing this, run the ltrace and strace command line applications and collect the output of the system calls and library calls that were used to interact with your file. Additionally, use the "perf stat <your program>" command to collect some performance statistics of your program. **Show all 3 outputs (ltrace, strace, perf) in your report.**

SOLUTION:
Ltrace output:

```
0.000000 SYS_brk(0)                                              = 0x556bab4cc000 <0.000034>
0.003657 SYS_access("/etc/ld.so.nohwcap", 00)                    = -2 <0.000105>
0.000198 SYS_access("/etc/ld.so.preload", 04)                    = -2 <0.000063>
0.000094 SYS_openat(0xffffff9c, 0x7faa8e5e0428, 0x80000, 0)      = 3 <0.000052>
0.000081 SYS_fstat(3, 0x7ffc2882e4f0)                            = 0 <0.000014>
0.000028 SYS_mmap(0, 0x13d16, 1, 2)                              = 0x7faa8e7d2000 <0.000029>
0.005238 SYS_close(3)                                            = 0 <0.000032>
0.000086 SYS_access("/etc/ld.so.nohwcap", 00)                    = -2 <0.000072>
0.000094 SYS_openat(0xffffff9c, 0x7faa8e7e8dd0, 0x80000, 0)      = 3 <0.000052>
0.000076 SYS_read(3, "\177ELF\002\001\001\003", 832)             = 832 <0.000015>
0.000060 SYS_fstat(3, 0x7ffc2882e550)                            = 0 <0.000014>
0.000028 SYS_mmap(0, 8192, 3, 34)                                = 0x7faa8e7d0000 <0.000029>
0.000060 SYS_mmap(0, 0x3f0ae0, 5, 2050)                          = 0x7faa8e1ce000 <0.000037>
0.000060 SYS_mprotect(0x7faa8e3b5000, 2097152, 0)                = 0 <0.000022>
0.000036 SYS_mmap(0x7faa8e5b5000, 0x6000, 3, 2066)               = 0x7faa8e5b5000 <0.000032>
0.000060 SYS_mmap(0x7faa8e5bb000, 0x3ae0, 3, 50)                 = 0x7faa8e5bb000 <0.000028>
0.000060 SYS_close(3)                                            = 0 <0.000014>
0.000045 SYS_arch_prctl(4098, 0x7faa8e7d1500, 0x7faa8e7d1e30, 0x7faa8e7d09b8) = 0 <0.000026>
0.000103 SYS_mprotect(0x7faa8e5b5000, 16384, 1)                  = 0 <0.000021>
0.000043 SYS_mprotect(0x556baa725000, 4096, 1)                   = 0 <0.000141>
0.000165 SYS_mprotect(0x7faa8e7e6000, 4096, 1)                   = 0 <0.000021>
0.000035 SYS_munmap(0x7faa8e7d2000, 81174)                       = 0 <0.000032>
0.014309 fopen("sanika.txt", "r" <unfinished ...>
0.000361     SYS_brk(0)                                          = 0x556bab4cc000 <0.000036>
0.000038     SYS_brk(0x556bab4ed000)                             = 0x556bab4ed000 <0.000015>
0.000039     SYS_openat(0xffffff9c, 0x556baa52546a, 0, 0)        = 3 <0.000041>
0.000071 <... fopen resumed> )                                   = 0x556bab4cc260 <0.000504>
0.000027 malloc(200)                                             = 0x556bab4cc490 <0.000101>
0.000134 printf("\nThe interesting string is %s\n", "sanika is fun " <unfinished ...>
0.000254     SYS_fstat(1, 0x7ffc2882e6d0)                        = 0 <0.000022>
0.000041     SYS_write(1, "\n", 1)                               = 1 <0.000081>
0.000106     SYS_write(1, "The interesting string is sanika"..., 41) = 41 <0.000301>
0.000333 <... printf resumed> )                                  = 42 <0.000727>
0.000026 fopen("sanika.txt", "w" <unfinished ...>
0.000293     SYS_openat(0xffffff9c, 0x556baa52546a, 577, 438)    = 4 <0.001549>
0.001595 <... fopen resumed> )                                   = 0x556bab4cc970 <0.001882>
0.000026 printf("%s\nfile has been created\n", "sanika.txt" <unfinished ...>
0.000243     SYS_write(1, "sanika.txt\nfile has been created"..., 33) = 33 <0.001534>
0.001590 <... printf resumed> )                                  = 33 <0.001817>
```

```
0.000102     SYS_write(5, "anika is fun sanika is fun yaayk"..., 56) = 56 <0.001535>
0.001566     SYS_close(5)                                        = 0 <0.000059>
0.000099 <... fclose resumed> )                                  = 0 <0.001759>
0.000029 fopen("sanika.txt", "a" <unfinished ...>
0.000155     SYS_openat(0xffffff9c, 0x556baa52546a, 1089, 438)   = 5 <0.000043>
0.000060     SYS_lseek(5, 0, 2)                                  = 56 <0.000013>
0.000031 <... fopen resumed> )                                   = 0x556bab4cdfc0 <0.000241>
0.000023 strlen("yaay")                                          = 4 <0.000104>
0.000135 fwrite("yaay", 1, 4, 0x556bab4cdfc0 <unfinished ...>
0.000119     SYS_fstat(5, 0x7ffc2882ecc0)                        = 0 <0.000022>
0.000035 <... fwrite resumed> )                                  = 4 <0.000148>
0.000140 fclose(0x556bab4cdfc0 <unfinished ...>
0.000122     SYS_write(5, "yaay", 4)                             = 4 <0.005048>
0.005101     SYS_close(5)                                        = 0 <0.000031>
0.000078 <... fclose resumed> )                                  = 0 <0.005285>
0.000032 puts("\nEnter the operation to be perfo"... <unfinished ...>
0.000297     SYS_write(1, "\n", 1)                               = 1 <0.001248>
0.001284     SYS_write(1, "Enter the operation to be perfor"..., 118) = 118 <0.003606>
0.003643 <... puts resumed> )                                    = 119 <0.005214>
0.000027 __isoc99_scanf(0x556baa52534b, 0x7ffc2882eea0, 0x7faa8e5bb8c0, 0x7faa8e2de154 <unfinished ...>
0.000309     SYS_read(0, "1\n", 1024)                            = 2 <1.913553>
1.913661 <... __isoc99_scanf resumed> )                          = 1 <1.913935>
0.000041 malloc(11)                                              = 0x556bab4cf200 <0.000266>
0.000303 __xstat(1, "sanika.txt", 0x7ffc2882ed90 <unfinished ...>
0.000150     SYS_stat("sanika.txt", 0x7ffc2882ed90)              = 0 <0.000054>
0.000068 <... __xstat resumed> )                                 = 0 <0.000213>
0.000023 printf("%s is the file permission\n", "rw-r--r--" <unfinished ...>
0.000223     SYS_write(1, "rw-r--r-- is the file permission"..., 33) = 33 <0.000486>
0.000514 <... printf resumed> )                                  = 33 <0.000730>
0.000026 puts("\nTo change the file permission o"... <unfinished ...>
0.000414     SYS_write(1, "\n", 1)                               = 1 <0.000104>
0.000093     SYS_write(1, "To change the file permission ou"..., 106) = 106 <0.000210>
0.000238 <... puts resumed> )                                    = 107 <0.000739>
0.000024 __isoc99_scanf(0x556baa52534b, 0x7ffc2882ee50, 0x7faa8e5bb8c0, 0x7faa8e2de154 <unfinished ...>
0.000372     SYS_read(0 <no return ...>
```

Strace output:

```
0.000000 execve("./hwq2", ["./hwq2", "sanika", "is", "fun"], 0x7ffc163fe508 /* 53 vars */) = 0 <0.000277>
0.004177 brk(NULL)                      = 0x56268ebe5000 <0.000007>
0.000299 access("/etc/ld.so.nohwcap", F_OK) = -1 ENOENT (No such file or directory) <0.000010>
0.000047 access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory) <0.000006>
0.000029 openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3 <0.000011>
0.000029 fstat(3, {st_mode=S_IFREG|0644, st_size=81174, ...}) = 0 <0.000006>
0.000026 mmap(NULL, 81174, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7ff0b2537000 <0.000009>
0.000026 close(3)                       = 0 <0.000005>
0.000025 access("/etc/ld.so.nohwcap", F_OK) = -1 ENOENT (No such file or directory) <0.000006>
0.000028 openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3 <0.000009>
0.000026 read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\260\34\2\0\0\0\0"..., 832) = 832 <0.000006>
0.000024 fstat(3, {st_mode=S_IFREG|0755, st_size=2030544, ...}) = 0 <0.000005>
0.000023 mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7ff0b2535000 <0.000007>
0.000026 mmap(NULL, 4131552, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7ff0b1f33000 <0.000008>
0.000024 mprotect(0x7ff0b211a000, 2097152, PROT_NONE) = 0 <0.000012>
0.000028 mmap(0x7ff0b231a000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e7000) = 0x7ff0b231a000 <0.000011>
0.000032 mmap(0x7ff0b2320000, 15072, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7ff0b2320000 <0.000007>
0.000032 close(3)                       = 0 <0.000005>
0.000032 arch_prctl(ARCH_SET_FS, 0x7ff0b2536500) = 0 <0.000004>
0.000072 mprotect(0x7ff0b231a000, 16384, PROT_READ) = 0 <0.000011>
0.000034 mprotect(0x56268e226000, 4096, PROT_READ) = 0 <0.000009>
0.000027 mprotect(0x7ff0b254b000, 4096, PROT_READ) = 0 <0.000010>
0.000027 munmap(0x7ff0b2537000, 81174) = 0 <0.000020>
0.003808 brk(NULL)                      = 0x56268ebe5000 <0.000009>
0.000040 brk(0x56268ec06000)            = 0x56268ec06000 <0.000008>
0.000032 openat(AT_FDCWD, "sanika.txt", O_RDONLY) = 3 <0.000011>
0.000041 fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 2), ...}) = 0 <0.000006>
0.000028 write(1, "\n", 1)              = 1 <0.000018>
0.000041 write(1, "The interesting string is sanika"..., 41) = 41 <0.000236>
0.000263 openat(AT_FDCWD, "sanika.txt", O_WRONLY|O_CREAT|O_TRUNC, 0666) = 4 <0.000052>
0.000087 write(1, "sanika.txt\nfile has been created"..., 33) = 33 <0.000018>
0.000037 fstat(4, {st_mode=S_IFREG|0644, st_size=0, ...}) = 0 <0.000005>
0.000028 write(4, "sanika is fun ", 14) = 14 <0.000026>
0.000057 close(4)                       = 0 <0.000154>
0.000188 openat(AT_FDCWD, "sanika.txt", O_WRONLY|O_CREAT|O_APPEND, 0666) = 4 <0.000010>
0.002230 lseek(4, 0, SEEK_END)          = 14 <0.000006>
0.000026 fstat(4, {st_mode=S_IFREG|0644, st_size=14, ...}) = 0 <0.000006>
0.000165 write(4, "yaay", 4)            = 4 <0.003362>
0.003711 close(4)                       = 0 <0.000020>
```

```
     0.000162 openat(AT_FDCWD, "sanika.txt", O_WRONLY|O_CREAT|O_APPEND, 0666) = 5 <0.000008>
     0.000029 lseek(5, 0, SEEK_END)     = 56 <0.000019>
     0.001301 fstat(5, {st_mode=S_IFREG|0644, st_size=56, ...}) = 0 <0.000177>
     0.000225 write(5, "yaay", 4)        = 4 <0.000012>
     0.000033 close(5)                  = 0 <0.000008>
     0.000026 write(1, "\n", 1)          = 1 <0.000053>
     0.000079 write(1, "Enter the operation to be perfor"..., 118) = 118 <0.000986>
     0.001025 read(0, "1\n", 1024)       = 2 <21.021059>
    21.021126 stat("sanika.txt", {st_mode=S_IFREG|0644, st_size=60, ...}) = 0 <0.000011>
     0.000042 write(1, "rw-r--r-- is the file permission"..., 33) = 33 <0.000009>
     0.000028 write(1, "\n", 1)          = 1 <0.000006>
     0.000026 write(1, "To change the file permission ou"..., 106) = 106 <0.000006>
     0.000024 read(0, 0x56268ebe5ba0, 1024) = ? ERESTARTSYS (To be restarted if SA_RESTART is set) <6.628532>
     6.628587 --- SIGINT {si_signo=SIGINT, si_code=SI_KERNEL} ---
     0.000107 +++ killed by SIGINT +++
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
  0.00    0.000000           0         9         1 read
  0.00    0.000000           0        33           write
  0.00    0.000000           0        10           close
  0.00    0.000000           0         1           stat
  0.00    0.000000           0        14           fstat
  0.00    0.000000           0         5           lseek
  0.00    0.000000           0         5           mmap
  0.00    0.000000           0         4           mprotect
  0.00    0.000000           0         1           munmap
  0.00    0.000000           0         3           brk
  0.00    0.000000           0         3         3 access
  0.00    0.000000           0         1           execve
  0.00    0.000000           0         1           arch_prctl
  0.00    0.000000           0        12           openat
------ ----------- ----------- --------- --------- ----------------
100.00    0.000000                   102         4 total
```

perf output:

```
# started on Sat Feb  9 12:21:48 2019


 Performance counter stats for './hwq2 sanika is fun':

          3.089502      task-clock (msec)         #    0.000 CPUs utilized
                26      context-switches          #    0.008 M/sec
                 0      cpu-migrations            #    0.000 K/sec
                54      page-faults               #    0.017 M/sec
   <not supported>      cycles
   <not supported>      instructions
   <not supported>      branches
   <not supported>      branch-misses

      26.444646459 seconds time elapsed
```

[Problem 3 - 30 Pts] Setup Buildroot, then build and boot a Beaglebone Green (BBG) Linux image

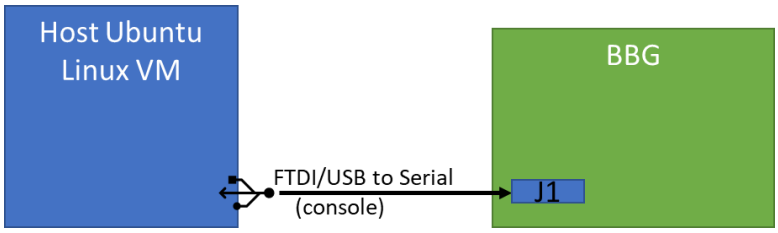
In this exercise you're building an micro SD memory card with everything to boot and run embedded Linux on a BBG. The images (uBoot, MLO, kernel, rootfs) for the BBG will be built using the Buildroot toolchain with MELP Chapter 6 as your guide.

(***Note***: We're moving on from Crosstools-NG to use the Buildroot Linux toolchain/development environment. As you begin, you'll want to make sure your Ubuntu (host) Linux VM has plenty of disk space (>100 GB). Secondly, over time, as you use install and setup the Buildroot tools, you'll want to adjust your dev environment (.bashrc, $PATH, $HOME/bin, etc) to access and use Buildroot's bin, etc.)

The process will involve:

- Setting up Buildroot along with the corresponding Linux source files (these are different files from your host Linux sources)
- configuring and building Buildroot toolchain
- making/configuring a target BBG Linux configuration (e.g. beaglebone_defconfig)
- building the Linux image

- burn the image files to your microSD memory card as a bootable file system
- set up your BBG HW
  - connect your dev host to BBG using a serial console cable for console interaction - With FTDI/USB serial port cable connected to your host and the other end to the BBG J1 connector (Black wire on cable is pin 1)



  - Download 'minicom' or 'screen' or 'gtkterm" or your favorite term emulator to your host.
    - `$ sudo screen /dev/ttyUSB0 115200`

    Or

    - `$ sudo apt-get install minicom`
    - `$ sudo minicom /dev/ttyUSB0 115200`

    Or

      - `$ sudo apt-get install gtkterm`
      - `$ sudo gtkterm -s 115200 -p /dev/ttyUSB0`
  - (Note: If you are using a guest Linux Virtual Machine (VM) for development on your host, there may be some configuration (mapping) needed of the FTDI/USB serial port from your host computer port through to your guest VM environment.)

- Install the micro SD memory card, and then apply power to the BBG . You may need to interrupt/change to normal BBG manufactured boot sequence to boot your image through the use of BBG buttons or console interaction …

- Change the BBG login greeting to include your name using make menuconfig and rebuild your image. Burn your new image to the microSD memory card. Congratulations, you've just build your own Linux embedded system!

Keep in mind the version of tools, kernel, etc of software mentioned in the book are newer now, so you'll have to be attentive to versions, paths, etc.

We're not going for any fancy functionality, but are just getting the tools set up, oriented to the location of files, and learning how to burn a micro SD memory card and get it to boot on the BBG. Please to not overwrite the eMMc memory that ships on the BBG **– you should only boot from your micro SD memory card**. (Hint: Maintaining the original manufacturers code on the BBG allows you to boot from the manufacturer's original image, by removing the micro SD memory card with your image, and see

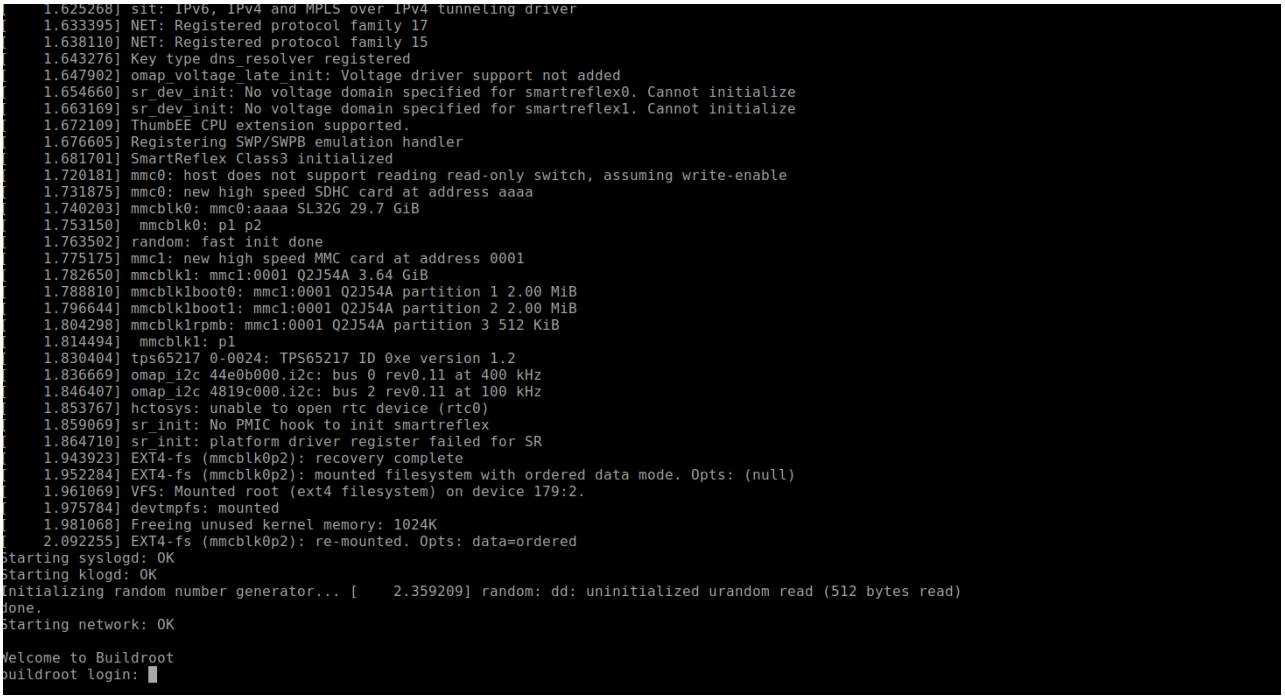
that the BBG is still functional. Good to know when you're wondering if it's a problem with your code or the HW!)

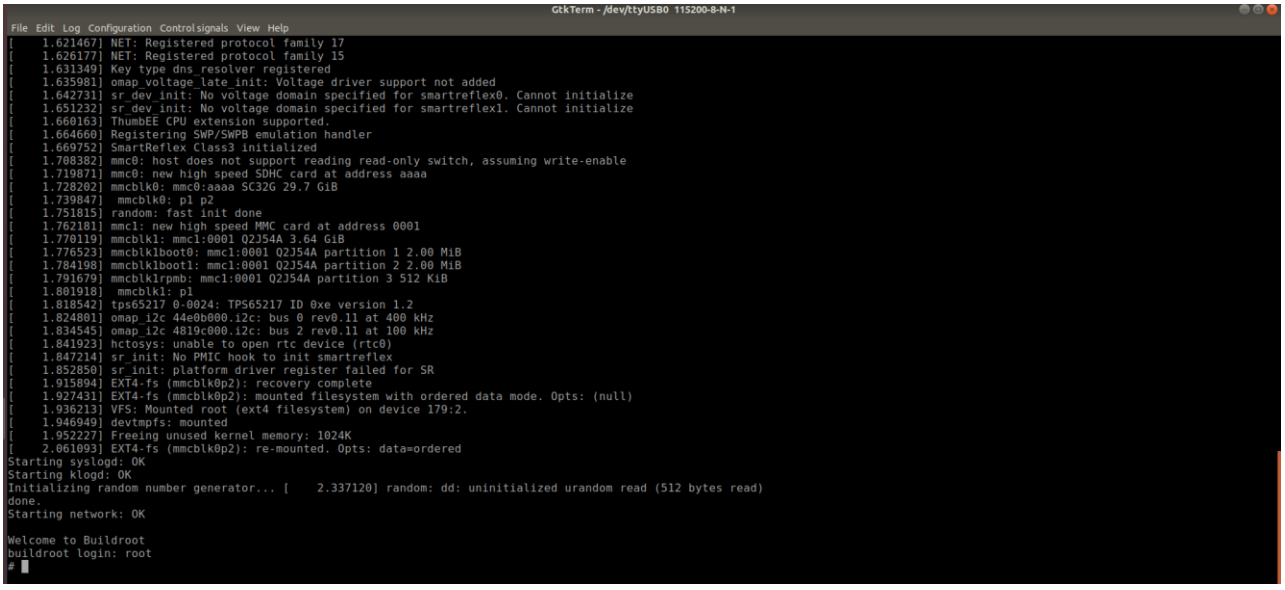
Boot your BBG, login to root, and try some commands on the console.

Report a screen capture of the last 20 lines of console boot sequence, as well as you logging in and executing your favorite commands at the console (e.g. ps -aux, lsmod, ls /).

SOLUTION:

Build root login:





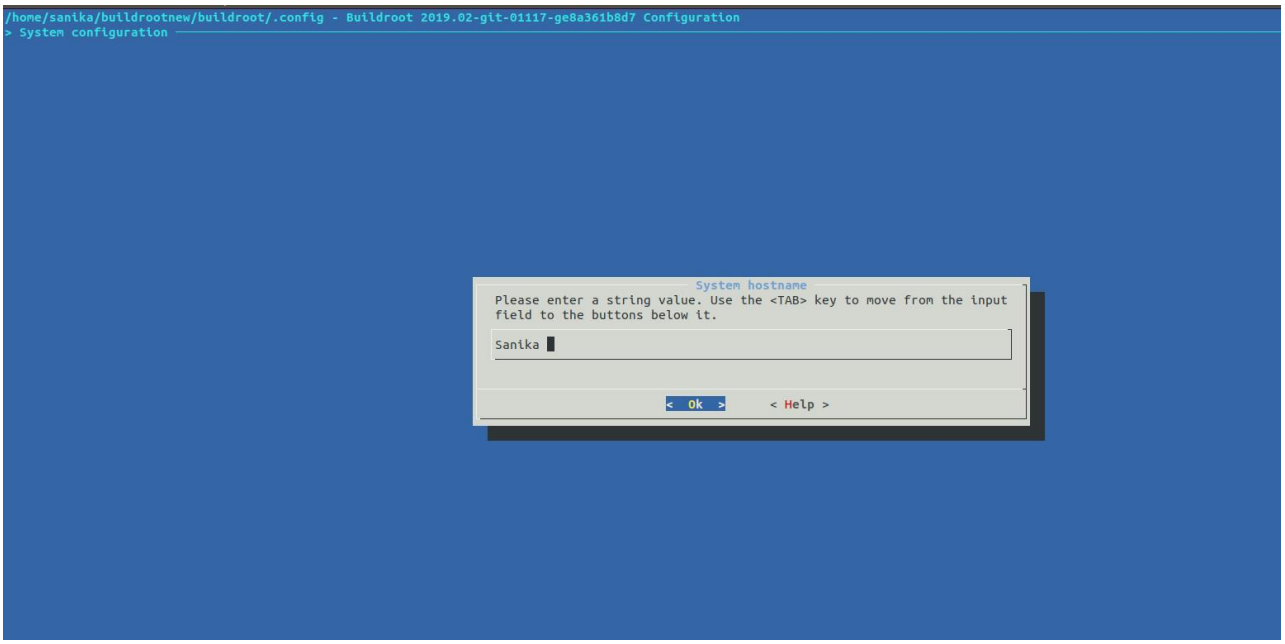
Boot sequence – 20 lines console:

```
[    2.061093] EXT4-fs (mmcblk0p2): re-mounted. Opts: data=ordered
Starting syslogd: OK
Starting klogd: OK
Initializing random number generator... [    2.337120] random: dd: uninitialized urandom read (512 bytes read)
done.
Starting network: OK

Welcome to Buildroot
buildroot login: root
# ls
# cd /
# ls
bin         lib         lost+found  opt         run         tmp
dev         lib32       media       proc        sbin        usr
etc         linuxrc     mnt         root        sys         var
# cd usr
# ls
bin    lib    lib32  sbin   share
# cd bin
# ls
[            eject        last         nslookup     shred        unlzop
[[           env          ldd          od           sort         unxz
ar           expr         less         openvt       strings      unzip
awk          factor       logger       passwd       svc          uptime
basename     fallocate    logname      paste        svok         uudecode
bunzip2      find         lsof         patch        tail         uuencode
bzcat        flock        lspci        printf       tee          vlock
chrt         fold         lsscsi       readlink     telnet       w
chvt         free         lsusb        realpath     test         wc
cksum        fuser        lzcat        renice       tftp         wget
clear        getconf      lzma         reset        time         which
cmp          head         lzopcat      resize       top          who
crontab      hexdump      md5sum       seq          tr           whoami
cut          hexedit      mesg         setfattr     traceroute   xargs
dc           hostid       microcom     setkeycodes  truncate     xxd
deallocvt    id           mkfifo       setsid       tty          xz
diff         install      mkpasswd     sha1sum      uniq         xzcat
dirname      ipcrm        nl           sha256sum    unix2dos     yes
dos2unix     ipcs         nohup        sha3sum      unlink
du           killall      nproc        sha512sum    unlzma
#
```

```
# ls -la
total 25
drwxr-xr-x   18 root     root          1024 Feb  8  2019 .
drwxr-xr-x   18 root     root          1024 Feb  8  2019 ..
drwxr-xr-x    2 root     root          2048 Feb  8  2019 bin
drwxr-xr-x    4 root     root          2780 Jan  1 00:00 dev
drwxr-xr-x    5 root     root          1024 Jan  1 00:00 etc
drwxr-xr-x    3 root     root          1024 Feb  8  2019 lib
lrwxrwxrwx    1 root     root             3 Feb  8  2019 lib32 -> lib
lrwxrwxrwx    1 root     root            11 Feb  8  2019 linuxrc -> bin/busybox
drwx------    2 root     root         12288 Feb  8  2019 lost+found
drwxr-xr-x    2 root     root          1024 Feb  7  2019 media
drwxr-xr-x    2 root     root          1024 Feb  7  2019 mnt
drwxr-xr-x    2 root     root          1024 Feb  7  2019 opt
dr-xr-xr-x   58 root     root             0 Jan  1 00:00 proc
drwx------    2 root     root          1024 Jan  1 00:02 root
drwxr-xr-x    3 root     root           140 Jan  1 00:00 run
drwxr-xr-x    2 root     root          1024 Feb  8  2019 sbin
dr-xr-xr-x   12 root     root             0 Jan  1 00:00 sys
drwxrwxrwt    2 root     root            60 Jan  1 00:00 tmp
drwxr-xr-x    6 root     root          1024 Feb  8  2019 usr
drwxr-xr-x    4 root     root          1024 Feb  8  2019 var
#
```

User login change:




```
[    1.669630] SmartReflex Class3 initialized
[    1.708244] mmc0: host does not support reading read-only switch, assuming write-enable
[    1.719566] mmc0: new high speed SDHC card at address aaaa
[    1.727895] mmcblk0: mmc0:aaaa SC32G 29.7 GiB
[    1.739772]  mmcblk0: p1 p2
[    1.751798] random: fast init done
[    1.763217] mmc1: new high speed MMC card at address 0001
[    1.770649] mmcblk1: mmc1:0001 Q2J54A 3.64 GiB
[    1.776769] mmcblk1boot0: mmc1:0001 Q2J54A partition 1 2.00 MiB
[    1.784604] mmcblk1boot1: mmc1:0001 Q2J54A partition 2 2.00 MiB
[    1.792256] mmcblk1rpmb: mmc1:0001 Q2J54A partition 3 512 KiB
[    1.802523]  mmcblk1: p1
[    1.818488] tps65217 0-0024: TPS65217 ID 0xe version 1.2
[    1.824747] omap_i2c 44e0b000.i2c: bus 0 rev0.11 at 400 kHz
[    1.834482] omap_i2c 4819c000.i2c: bus 2 rev0.11 at 100 kHz
[    1.841836] hctosys: unable to open rtc device (rtc0)
[    1.847136] sr_init: No PMIC hook to init smartreflex
[    1.852779] sr_init: platform driver register failed for SR
[    1.916056] EXT4-fs (mmcblk0p2): recovery complete
[    1.924677] EXT4-fs (mmcblk0p2): mounted filesystem with ordered data mode. Opts: (null)
[    1.933471] VFS: Mounted root (ext4 filesystem) on device 179:2.
[    1.944343] devtmpfs: mounted
[    1.949625] Freeing unused kernel memory: 1024K
[    2.062360] EXT4-fs (mmcblk0p2): re-mounted. Opts: data=ordered
Starting syslogd: OK
Starting klogd: OK
Initializing random number generator... [    2.334951] random: dd: uninitialized urandom read (512 bytes read)
done.
Starting network: OK

HI
sanika login: root
# pwd
/root
# ls
# cd /
# ls
bin         lib         lost+found  opt         run         tmp
dev         lib32       media       proc        sbin        usr
etc         linuxrc     mnt         root        sys         var
#
```

```
sanika login: root
# pwd
/root
# ls
# cd /
# ls
bin         lib         lost+found  opt         run         tmp
dev         lib32       media       proc        sbin        usr
etc         linuxrc     mnt         root        sys         var
# ls -l
total 23
drwxr-xr-x    2 root     root          2048 Feb  8  2019 bin
drwxr-xr-x    4 root     root          2780 Jan  1 00:00 dev
drwxr-xr-x    5 root     root          1024 Jan  1 00:00 etc
drwxr-xr-x    3 root     root          1024 Feb  8  2019 lib
lrwxrwxrwx    1 root     root             3 Feb  8  2019 lib32 -> lib
lrwxrwxrwx    1 root     root            11 Feb  8  2019 linuxrc -> bin/busybox
drwx------    2 root     root         12288 Feb  8  2019 lost+found
drwxr-xr-x    2 root     root          1024 Feb  7  2019 media
drwxr-xr-x    2 root     root          1024 Feb  7  2019 mnt
drwxr-xr-x    2 root     root          1024 Feb  7  2019 opt
dr-xr-xr-x   58 root     root             0 Jan  1 00:00 proc
drwx------    2 root     root          1024 Jan  1 00:02 root
drwxr-xr-x    3 root     root           140 Jan  1 00:00 run
drwxr-xr-x    2 root     root          1024 Feb  8  2019 sbin
dr-xr-xr-x   12 root     root             0 Jan  1 00:00 sys
drwxrwxrwt    2 root     root            60 Jan  1 00:00 tmp
drwxr-xr-x    6 root     root          1024 Feb  8  2019 usr
drwxr-xr-x    4 root     root          1024 Feb  8  2019 var
#
```

```
lrwxrwxrwx    1 root     root            17 Feb  7  2019 vlock -> ../../bin/busybox
lrwxrwxrwx    1 root     root            17 Feb  7  2019 w -> ../../bin/busybox
lrwxrwxrwx    1 root     root            17 Feb  7  2019 wc -> ../../bin/busybox
lrwxrwxrwx    1 root     root            17 Feb  7  2019 wget -> ../../bin/busybox
lrwxrwxrwx    1 root     root            17 Feb  7  2019 which -> ../../bin/busybox
lrwxrwxrwx    1 root     root            17 Feb  7  2019 who -> ../../bin/busybox
lrwxrwxrwx    1 root     root            17 Feb  7  2019 whoami -> ../../bin/busybox
lrwxrwxrwx    1 root     root            17 Feb  7  2019 xargs -> ../../bin/busybox
lrwxrwxrwx    1 root     root            17 Feb  7  2019 xxd -> ../../bin/busybox
lrwxrwxrwx    1 root     root            17 Feb  7  2019 xz -> ../../bin/busybox
lrwxrwxrwx    1 root     root            17 Feb  7  2019 xzcat -> ../../bin/busybox
lrwxrwxrwx    1 root     root            17 Feb  7  2019 yes -> ../../bin/busybox
# pwd
/usr/bin
# cd /
# pwd
/
#
```

[Problem 4 - 10 Pts] Port Your File IO program to BBG

Now that you have a cross development environment setup, let's port your program from Problem 1 to the BBG. The simplest way to do this is to cross-compile your program "out-of-tree" in some project directory outside of the Buildroot directories. (e.g. ~/projects/myProblem4).

Then create an overlay directory inside the Buildroot directory –

(e.g. ~/buildroot/board/beaglebone_rick/root-overlay/usr/bin) and copy your "arm compiled" executable there. Next configure Buildroot (make menuconfig) to add an overlay directory that will deliver your file (add to) the rootfs in next complete image you build and burn to the micro SD memory. This will put your executable in a good location is in the (target) BBG filesystem's /usr/bin directory.

Additionally, configure Buildroot to build/add to your BBG image the strace, ltrace, and perf executables using "make menuconfig" to include the correct packages (e.g. for ltrace search for "ltrace" then set symbol: BR2_PACKAGE_LTRACE [=y]).

Remake, burn a new complete micro SD memory card image, install and boot/run your new Linux image.

From a BBG console command line, run your program and collect the BBG version of ltrace, strace and perf stat of the interesting stats that you collected from Problem 1.

SOLUTION:

Ltrace:

```
# ltrace ./app hello
dwfl_report_elf app@0x10000 (/app) 104: address range overlaps an existing module
Backend initialization failed.
Couldn't load ELF object /lib/ld-uClibc.so.0: Success
__uClibc_main(0x10c70, 2, 0xbef17e54, 0x10518 <unfinished ...>
fopen("sanika.txt", "r")                          = 0x23008
malloc(200)                                       = 0x24058
fopen("sanika.txt", "w")                          = 0x24128
printf("%s\nfile has been created\n", "sanika.txt"sanika.txt
file has been created
) = 33
strlen("hello ")                                  = 6
fwrite("hello ", 1, 6, 0x24128)                    = 6
fclose(0x24128)                                   = 0
fopen("sanika.txt", "a")                          = 0x24128
strlen("yaay")                                    = 4
fwrite("yaay", 1, 4, 0x24128)                     = 4
fclose(0x24128)                                   = 0
puts("\nEnter the operation to be perfo"...
Enter the operation to be performed: 1.Modify permission of file, 2.flush file, 3.reading character, 4.reading string
)     = 119
scanf(0x10fdc, 0xbef17ce4, 1, 0xbef17ce42
)        = 1
fopen("sanika.txt", "a")                          = 0x24128
fwrite(" HI I am sanika", 1, 15, 0x24128)         = 15
fflush(0x24128)                                   = 0
puts("File output flush complete"File output flush complete
)               = 27
fopen("sanika.txt", "w")                          = 0x25178
printf("%s\nfile has been created\n", "sanika.txt"sanika.txt
file has been created
) = 33
strlen("hello hello ")                            = 12
fwrite("hello hello ", 1, 12, 0x25178)            = 12
```

```
fopen("sanika.txt", "a")                          = 0x25178
strlen("yaay")                                    = 4
fwrite("yaay", 1, 4, 0x25178)                     = 4
fclose(0x25178)                                   = 0
puts("\nEnter the operation to be perfo"...
Enter the operation to be performed: 1.Modify permission of file, 2.flush file, 3.reading character, 4.reading string
)     = 119
scanf(0x10fdc, 0xbef17ce4, 1, 0xbef17ce43
)        = 1
fgetc(0x23008)                                    = 'h'
printf("The character read is %c", 'h')           = 23
fopen("sanika.txt", "w")                          = 0x25178
printf("%s\nfile has been created\n", "sanika.txt"The character read is hsanika.txt
file has been created
) = 33
strlen("hello hello hello ")                      = 18
fwrite("hello hello hello ", 1, 18, 0x25178)      = 18
fclose(0x25178)                                   = 0
fopen("sanika.txt", "a")                          = 0x25178
strlen("yaay")                                    = 4
fwrite("yaay", 1, 4, 0x25178)                     = 4
fclose(0x25178)                                   = 0
puts("\nEnter the operation to be perfo"...
Enter the operation to be performed: 1.Modify permission of file, 2.flush file, 3.reading character, 4.reading string
)     = 119
scanf(0x10fdc, 0xbef17ce4, 1, 0xbef17ce44
```

```
printf(" %s ", "ello hello yaayo yaay")          = 23
fopen("sanika.txt", "w")                         = 0x25178
printf("%s\nfile has been created\n", "sanika.txt" ello hello yaayo yaay sanika.txt
file has been created
) = 33
strlen("ello hello yaayo yhello ")               = 24
fwrite("ello hello yaayo yhello ", 1, 24, 0x25178) = 24
fclose(0x25178)                                  = 0
fopen("sanika.txt", "a")                         = 0x25178
strlen("yaay")                                   = 4
fwrite("yaay", 1, 4, 0x25178)                    = 4
fclose(0x25178)                                  = 0
puts("\nEnter the operation to be perfo"...
Enter the operation to be performed: 1.Modify permission of file, 2.flush file, 3.reading character, 4.reading string
)      = 119
scanf(0x10fdc, 0xbef17ce4, 1, 0xbef17ce4^C <no return ...>
--- SIGINT (Interrupt) ---
+++ killed by SIGINT +++
# 
```

Strace:

```
# strace ./app hello its me
execve("./app", ["./app", "hello", "its", "me"], 0xbefa0e3c /* 12 vars */) = 0
readlinkat(AT_FDCWD, "/proc/self/exe", "/app", 4096) = 4
mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb6fc8000
open("/lib//libc.so.0", O_RDONLY)       = 3
fstat(3, {st_mode=S_IFREG|0755, st_size=440148, ...}) = 0
mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb6fc7000
read(3, "\177ELF\1\1\1\0\0\0\0\0\0\0\0\0\3\0(\0\1\0\0\0@\357\0\0004\0\0\0"..., 4096) = 4096
mmap2(NULL, 598016, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb6f22000
mmap2(0xb6f22000, 435932, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED, 3, 0) = 0xb6f22000
mmap2(0xb6f9c000, 4852, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED, 3, 0x6a000) = 0xb6f9c000
mmap2(0xb6f9e000, 88104, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0xb6f9e000
close(3)                                = 0
munmap(0xb6fc7000, 4096)                = 0
stat("/lib/ld-uClibc.so.0", {st_mode=S_IFREG|0755, st_size=28516, ...}) = 0
mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb6fc7000
set_tls(0xb6fc7490)                     = 0
mprotect(0x21000, 4096, PROT_READ)      = 0
mprotect(0xb6f9c000, 4096, PROT_READ)   = 0
mprotect(0xb6fc9000, 4096, PROT_READ)   = 0
set_tid_address(0xb6fc7068)             = 96
set_robust_list(0xb6fc706c, 12)         = 0
rt_sigaction(SIGRTMIN, {sa_handler=0xb6f80904, sa_mask=[], sa_flags=SA_RESTORER|SA_SIGINFO, sa_restorer=0xb6f31920}, NULL, 8) = 0
rt_sigaction(SIGRT_1, {sa_handler=0xb6f809c8, sa_mask=[], sa_flags=SA_RESTORER|SA_RESTART|SA_SIGINFO, sa_restorer=0xb6f31920}, NULL, 8) = 0
rt_sigprocmask(SIG_UNBLOCK, [RTMIN RT_1], NULL, 8) = 0
ugetrlimit(RLIMIT_STACK, {rlim_cur=8192*1024, rlim_max=RLIM_INFINITY}) = 0
ioctl(0, TCGETS, {B115200 opost isig icanon echo ...}) = 0
ioctl(1, TCGETS, {B115200 opost isig icanon echo ...}) = 0
brk(NULL)                               = 0x23000
brk(0x24000)                            = 0x24000
```

```
write(5, "yaay", 4)                     = 4
close(5)                                = 0
write(1, "\nEnter the operation to be perfo"..., 118
Enter the operation to be performed: 1.Modify permission of file, 2.flush file, 3.reading character, 4.reading string) = 118
write(1, "\n", 1
)                       = 1
read(0, 1
"1\n", 4096)                    = 2
stat64("sanika.txt", {st_mode=S_IFREG|0644, st_size=56, ...}) = 0
write(1, "rw-r--r-- is the file permission"..., 33rw-r--r-- is the file permission
) = 33
write(1, "\nTo change the file permission o"..., 106
To change the file permission out of: r, w,rw,no choose them as '1', '2', '3' or '4' options respectively) = 106
write(1, "\n", 1
)                       = 1
read(0, 3
"3\n", 4096)                    = 2
write(1, "To change the file permission fo"..., 110To change the file permission for groups to r, w, wr choose them as '1', '2', '3' or '4' options respectively
) = 110
read(0, 1
"1\n", 4096)                    = 2
write(1, "\nTo change the permission for ot"..., 36
To change the permission for others) = 36
write(1, "\n", 1
)                       = 1
read(0, 1
"1\n", 4096)                    = 2
chmod("sanika.txt", 0116764)            = 0
stat64("sanika.txt", {st_mode=S_IFREG|S_ISUID|S_ISGID|0764, st_size=56, ...}) = 0
write(1, "\nAfter changing the permission i"..., 38
After changing the permission is now ) = 38
write(1, "rwxrw-r--\n", 10rwxrw-r--
)           = 10
--- SIGSEGV {si_signo=SIGSEGV, si_code=SEGV_MAPERR, si_addr=0x17674} ---
+++ killed by SIGSEGV +++
```

Perf:

```
# perf stat ./app hello its me
sanika.txt
file has been created

Enter the operation to be performed: 1.Modify permission of file, 2.flush file, 3.reading character, 4.reading string
2
File output flush complete
sanika.txt
file has been created

Enter the operation to be performed: 1.Modify permission of file, 2.flush file, 3.reading character, 4.reading string
3
The character read is hsanika.txt
file has been created

Enter the operation to be performed: 1.Modify permission of file, 2.flush file, 3.reading character, 4.reading string
4
 ello its me hello its me yaayo its me yaay sanika.txt
file has been created

Enter the operation to be performed: 1.Modify permission of file, 2.flush file, 3.reading character, 4.reading string
1
rwxr--r-- is the file permission

To change the file permission out of: r, w,rw,no choose them as '1', '2', '3' or '4' options respectively
3
To change the file permission for groups to r, w, wr choose them as '1', '2', '3' or '4' options respectively
1

To change the permission for others
1

After changing the permission is now rw-rw-r--
```

The perf stats for the file handling operations code is:

```
Performance counter stats for './app hello its me':

         19.996667      task-clock (msec)         #    0.000 CPUs utilized
                24      context-switches          #    0.001 M/sec
                 0      cpu-migrations            #    0.000 K/sec
                32      page-faults               #    0.002 M/sec
          18031820      cycles                    #    0.902 GHz
           4397147      instructions              #    0.24  insn per cycle
            484580      branches                  #   24.233 M/sec
            124929      branch-misses             #   25.78% of all branches

      41.576629297 seconds time elapsed

# 
```

[Problem 5 - 30 Pts] Implement Your Own System Call on BBG

You are to create your own system call that can sort an array of numbers in kernel mode. This is more for practice of implementing a call than for making something for its utility. This system call needs to support the following features:

- A set of input parameters from user space including
  - Pointer to a buffer (input)
  - Size of that buffer (entries or bytes)
  - Pointer to a sorted buffer
- Validation of all input parameters
- Print information to the kernel buffer (log)
  - Log the input (user space) buffer contents when your syscall enters, exits, the size of the buffer,
  - Log the output buffer at start/completion of the sort (details provided below)
- Your system call needs to allocate dynamic memory to copy data in from user space
  - The user space array needs to be copied into kernel space presort
  - The array needs to be copied back to user space post sort.

Your syscall should be defined appropriately using the SYSCALL_DEFINE macro given your argument list size. The input buffer should be at least 256 int32_t data items. The buffer needs to be copied into

kernel space. Your syscall will need to sort the data in an order from largest to smallest. Once sorted, it needs to pass the sorted data back to the calling user application in a sorted buffer array.

Review the System Call lecture for additional guidance on where to add/how to add your code. In other words, add your syscall code to the kernel image built by Buildroot's make. Keep in mind, we're building a Beaglebone (arm-based) image, so some source modifications will be in the Buildroot kernel source directory (e.g. ~/buildroot/output/build/linux-<something>/kernel) while other source code modifications will be in architecture specific directories (e.g. ~/buildroot/output/build/linux-<something>/arch/arm…).

Show that your kernel module works by writing a user space application that calls your system call numerous times. You can randomly generate this buffer of data elements using random() and time(). This application can be built "out-of-tree" from Buildroot but you should use (include) Buildroot Linux directories of source files (i.e. ~/buildroot/output/build/linux-headers-<xxx> and ~/buildroot/output/build/linux-<xxx>" used to compile the BBG kernel.

You should show that:

- System call works correctly (all input parameters valid and correct)
    - Print information showing that the sort worked correctly
- System call fails (input parameters are not valid and/or correct)
    - BONUS: All errors should return appropriate error values (defined in errno.h and errno-base.h)
- Bonus: use a sort algorithm with O(nLogn) performance

Your report should include screenshots of example output of your multiple program runs, including both successful and failure cases (annotated with the "use-case" comments), appropriate BBG kernel logs with printk comments noting the use-cases. Report the time it takes to perform your syscall using timestamps from the log files.

SOLUTION:

Sorting using bubble sort:

```
# ./sortfunc.elf
Before sorting
137, 104, 136, 58, 55, 135, 89, 81, 140, 100, 49, 112, 49, 78, 121, 105, 118, 11
5, 79, 76, 95, 68, 95, 66, 130, 115, 50, 122, 65, 126, 67, 59, 54, 59, 69, 62, 5
0, 110, 95, 111, 67, 64, 79, 132, 62, 121, 61, 100, 60, 92, 129, 107, 80, 48, 12
5, 66, 83, 95, 140, 100, 142, 127, 111, 52, 138, 101, 130, 109, 131, 50, 76, 118
, 66, 107, 106, 49, 52, 119, 101, 64, 132, 54, 123, 68, 118, 73, 55, 57, 88, 51,
 78, 86, 99, 109, 59, 93, 66, 141, 58, 54, 143, 54, 92, 130, 82, 55, 99, 86, 94,
 56, 103, 50, 127, 82, 135, 69, 75, 142, 143, 116, 113, 77, 122, 68, 106, 133, 1
14, 125, 99, 92, 99, 66, 99, 143, 116, 133, 118, 71, 139, 133, 48, 98, 135, 95,
101, 94, 116, 128, 92, 115, 68, 62, 112, 143, 82, 75, 100, 116, 120, 55, 65, 139
, 138, 84, 138, 110, 137, 81, 134, 132, 134, 134, 55, 93, 85, 108, 140, 121, 60,
 88, 93, 49, 102, 125, 48, 105, 120, 68, 77, 96, 76, 94, 91, 70, 98, 54, 132, 91
, 55, 90, 80, 141, 80, 87, 90, 85, 115, 86, 63, 95, 95, 76, 96, 53, 57, 64, 78,
98, 85, 108, 50, 81, 122, 62, 103, 77, 132, 59, 120, 139, 102, 120, 104, 102, 12
7, 50, 108, 98, 57, 123, 50, 104, 55, 66, 77, 128, 83, 108, 82, 120, 136, 53, 12
1, 114, 67, 80,
After sorting the array is
143, 143, 143, 143, 142, 142, 141, 141, 140, 140, 140, 139, 139, 139, 138, 138,
138, 137, 137, 136, 136, 135, 135, 135, 134, 134, 134, 133, 133, 133, 132, 132,
132, 132, 132, 131, 130, 130, 130, 129, 128, 128, 127, 127, 127, 126, 125, 125,
125, 123, 123, 122, 122, 122, 121, 121, 121, 121, 120, 120, 120, 120, 120, 119,
118, 118, 118, 118, 116, 116, 116, 116, 115, 115, 115, 115, 114, 114, 113, 112,
112, 111, 111, 110, 110, 109, 109, 108, 108, 108, 108, 107, 107, 106, 106, 105,
105, 104, 104, 104, 103, 103, 102, 102, 102, 101, 101, 101, 100, 100, 100, 100,
99, 99, 99, 99, 99, 98, 98, 98, 96, 96, 95, 95, 95, 95, 95, 95, 94, 94,
94, 93, 93, 93, 92, 92, 92, 92, 91, 91, 90, 90, 89, 88, 88, 87, 86, 86, 86, 85,
85, 85, 84, 83, 83, 82, 82, 82, 82, 81, 81, 81, 80, 80, 80, 80, 79, 79, 78, 78,
78, 77, 77, 77, 77, 76, 76, 76, 76, 75, 75, 73, 71, 70, 69, 69, 68, 68, 68, 68,
68, 67, 67, 67, 66, 66, 66, 66, 66, 66, 65, 65, 64, 64, 64, 63, 62, 62, 62, 62,
61, 60, 60, 59, 59, 59, 59, 58, 58, 57, 57, 57, 56, 55, 55, 55, 55, 55, 55, 55,
54, 54, 54, 54, 54, 53, 53, 52, 52, 51, 50, 50, 50, 50, 50, 50, 50, 49, 49, 49,
49, 48, 48, 48,
sorting gets done
#
```

Failed use cases: The use case for checking errors - passed

```
# ./sortfunctest.elf
Before sorting
146, 118, 103, 95, 124, 146, 135[   18.215670] System call entrance
, 100, 120, 145, 94, 49, 70, 146, 70, 111, 118, [   18.222274] Data copying error from user
98, 49, 131, 92, 63, 79, 107, 96, 139, 87, 53, 6[   18.231613] System call entrance
2, 88, 52, 61, 59, 60, 60, 87, 110, 100, 91, 82,[   18.238323] Data copying error from user
 97, 137, 135, 71, 87, 57, 134, 57, 60, 88, 140,[   18.246745] System call entrance
 104, 55, 123, 116, 103, 115, 55, 61, 129, 96, 6[   18.254177] ------------[ cut here ]------------
5, 94, 59, 129, 107, 98, 91, 59, 141, 77, 60, 82[   18.263205] WARNING: CPU: 0 PID: 97 at mm/slab_common.c:971 kmalloc_slab+0x88/0xa4
, 65, 83, 121, 126, 122, 130, 138, 62, 122, 147,[   18.275302] Modules linked in:
 121, 50, 115, 77, 117, 122, 90, 98, 122, 59, 14[   18.282724] CPU: 0 PID: 97 Comm: sortfunctest.el Not tainted 4.14.40 #10
[   18.293863] Hardware name: Generic AM33XX (Flattened Device Tree)
[   18.300258] [<c0110b78>] (unwind_backtrace) from [<c010cc04>] (show_stack+0x10/0x14)
[   18.308364] [<c010cc04>] (show_stack) from [<c0832c44>] (dump_stack+0xb0/0xe8)
[   18.315921] [<c0832c44>] (dump_stack) from [<c0137998>] (__warn+0xd8/0x104)
[   18.323195] [<c0137998>] (__warn) from [<c0137a70>] (warn_slowpath_null+0x20/0x28)
[   18.331110] [<c0137a70>] (warn_slowpath_null) from [<c0285314>] (kmalloc_slab+0x88/0xa4)
[   18.339572] [<c0285314>] (kmalloc_slab) from [<c02ab784>] (__kmalloc+0x18/0x2f0)
[   18.347314] [<c02ab784>] (__kmalloc) from [<c014f134>] (SyS_sortfunc+0x20/0xdc)
[   18.354963] [<c014f134>] (SyS_sortfunc) from [<c0107ea0>] (ret_fast_syscall+0x0/0x28)
5, 133, 93, 56, 135, 136, 67, 80, 66, 131, 114, [   18.363289] ---[ end trace 3be64d74164eb28c ]---
135, 66, 87, 113, 92, 69, 56, 106, 96, 107, 132,[   18.372293] The malloc for buffk failed
 98, 74, 61, 67, 100, 55, 69, 75, 118, 118, 60, 63, 126, 100, 104, 97, 132, 122,[   18.384014] System call entrance
 80, 99, 61, 51, 138, 78, 95, 112, 138, 106, 60,[   18.391071] Data copying error from user
 97, 90, 62, 75, 55, 81, 128, 114, 54, 107, 84, 125, 71, 52, 55, 123, 108, 105, 108, 134, 89, 111, 147, 144, 101, 77, 92, 65, 68, 102, 129, 69, 96, 143, 97, 55, 128, 129, 121, 135, 140,
 109, 64, 63, 113, 71, 91, 73, 80, 103, 59, 122, 66, 110, 118, 71, 140, 114, 141, 112, 120, 122, 85, 68, 118, 134, 75, 50, 67, 100, 137, 59, 62, 105, 127, 127, 129, 70, 53, 113, 77, 116
, 87, 147, 79, 110, 70, 123, 76, 63, 139, 49, 90, 76, 121, 112, 115, 101, 114, 134, 53, 104, 146, 67, 113, 77, 99, 94, 99, 56, 60, 80,
The sorted output is NULL: Bad address
The length is less than 0: Cannot allocate memory
The length is less than 256: Bad address
System call has been failed -1
#
```

Dmesg for bad use cases:

```
[   18.215670] System call entrance
[   18.222274] Data copying error from user
[   18.231613] System call entrance
[   18.238323] Data copying error from user
[   18.246745] System call entrance
[   18.254177] ------------[ cut here ]------------
[   18.263205] WARNING: CPU: 0 PID: 97 at mm/slab_common.c:971 kmalloc_slab+0x88/0xa4
[   18.275302] Modules linked in:
[   18.282724] CPU: 0 PID: 97 Comm: sortfunctest.el Not tainted 4.14.40 #10
[   18.293863] Hardware name: Generic AM33XX (Flattened Device Tree)
[   18.300258] [<c0110b78>] (unwind_backtrace) from [<c010cc04>] (show_stack+0x10/0x14)
[   18.308364] [<c010cc04>] (show_stack) from [<c0832c44>] (dump_stack+0xb0/0xe8)
[   18.315921] [<c0832c44>] (dump_stack) from [<c0137998>] (__warn+0xd8/0x104)
[   18.323195] [<c0137998>] (__warn) from [<c0137a70>] (warn_slowpath_null+0x20/0x28)
[   18.331110] [<c0137a70>] (warn_slowpath_null) from [<c0285314>] (kmalloc_slab+0x88/0xa4)
[   18.339572] [<c0285314>] (kmalloc_slab) from [<c02ab784>] (__kmalloc+0x18/0x2f0)
[   18.347314] [<c02ab784>] (__kmalloc) from [<c014f134>] (SyS_sortfunc+0x20/0xdc)
[   18.354963] [<c014f134>] (SyS_sortfunc) from [<c0107ea0>] (ret_fast_syscall+0x0/0x28)
[   18.363289] ---[ end trace 3be64d74164eb28c ]---
[   18.372293] The malloc for buffk failed
[   18.384014] System call entrance
[   18.391071] Data copying error from user
# 
```

[Problem 6 - 10 Pts] Create a CRON/Systemd task on BBG

Write a C-program that uses your system call from problem 5 along with a few other system calls listed below. This program should run every 10 minutes and it should run to completion. Your program should collect the following information using system call APIs and print its output to a file (either write to the file or just redirect the output):

- Current Process ID
- Current User ID
- Current date and time
- Output of your system call

Report the collected information and the outputs from your system call for a period greater than 30 minutes.

```
# crontab -l
*/10 * * * * cd ~/../ && ./sortfunc.elf

#
```