

ECE4730J

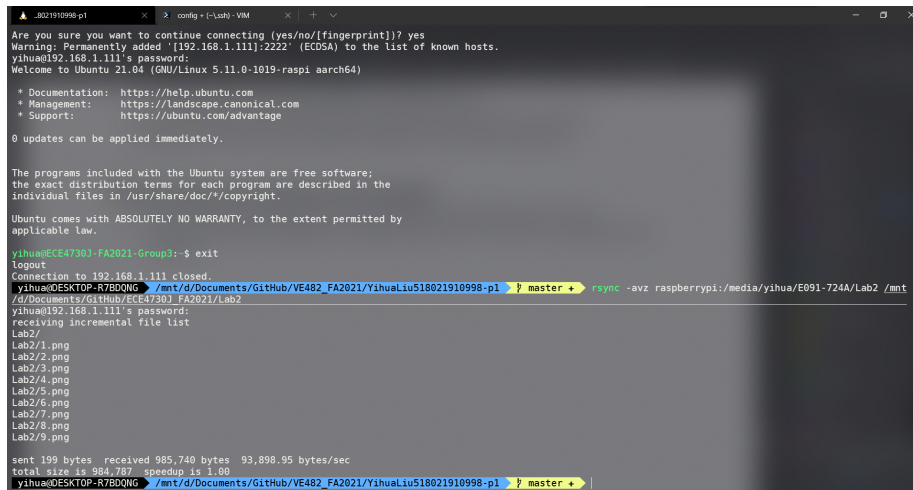
Advanced Embedded System

LAB 2: PROGRAM PROFILING ON LINUX

October 11, 2021

Name	Student ID
Yihua Liu	518021910998
Shuocheng Chen	517021911139
Yiming Ju	518370910059

Lab 1 Task 3. (Optional)



```
J801910998-p1  x  config - (-Lab2) - VM  x  +  x
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[192.168.1.111]:2222' (ECDSA) to the list of known hosts.
yihua@192.168.1.111's password:
Welcome to Ubuntu 21.04 (GNU/Linux 5.11.0-1019-raspi aarch64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

0 updates can be applied immediately.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

yihua@ECE4730J-FA2021-Group3:~$ exit
logout
Connection to 192.168.1.111 closed.
yihua@DESKTOP-R7BDQNG ~$ /mnt/d/Documents/GitHub/VE482_FA2021/YihuaLiu518021910998-p1  master +  rsync -avz raspberryypi:/media/yihua/E091-724A/Lab2 /mnt/d/Documents/GitHub/ECE4730J_FA2021/Lab2
yihua@192.168.1.111's password:
receiving incremental file list
Lab2/
Lab2/1.png
Lab2/2.png
Lab2/3.png
Lab2/4.png
Lab2/5.png
Lab2/6.png
Lab2/7.png
Lab2/8.png
Lab2/9.png
sent 199 bytes  received 985,740 bytes  93,898.95 bytes/sec
total size is 984,787  speedup is 1.00
yihua@DESKTOP-R7BDQNG ~$ /mnt/d/Documents/GitHub/VE482_FA2021/YihuaLiu518021910998-p1  master +
```

Figure 1. Setup the remote access to your Raspberry Pi.

1. No submission for the first exercise.
- 2.

```
yihua@ECE4730J-FA2021-Group3: ~/test_programs
yihua@ECE4730J-FA2021-Group3:~/test_programs$ gcc arr_search.c -o arr_search_dynamic -O0 -lm
yihua@ECE4730J-FA2021-Group3:~/test_programs$ gcc arr_search.c -o arr_search_static -O0 -static -lm
yihua@ECE4730J-FA2021-Group3:~/test_programs$ ls -lh
total 648K
-rw-r--r-- 1 yihua yihua 2.6K 8月 15 2020 arr_search.c
-rwxrwxr-x 1 yihua yihua 14K 10月 11 20:08 arr_search_dynamic
-rwxrwxr-x 1 yihua yihua 607K 10月 11 20:08 arr_search_static
-rw-r--r-- 1 yihua yihua 1.9K 1月 25 2016 dense_mm.c
-rw-r--r-- 1 yihua yihua 201 1月 25 2016 Makefile
-rw-r--r-- 1 yihua yihua 2.8K 9月 16 2015 parallel_dense_mm.c
-rw-r--r-- 1 yihua yihua 1.1K 1月 25 2016 sing.c
-rw-r--r-- 1 yihua yihua 3.0K 9月 16 2015 sort.c
yihua@ECE4730J-FA2021-Group3:~/test_programs$
```

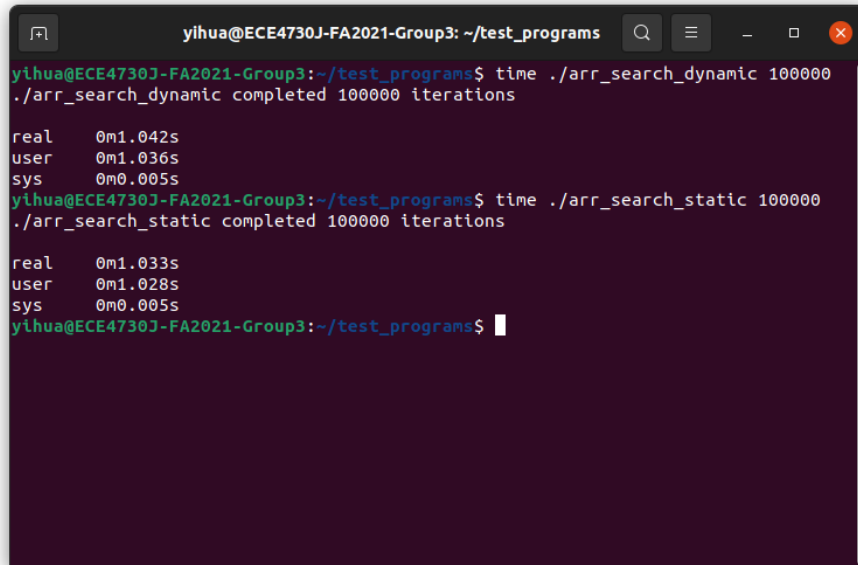
Figure 2. The list of the size of each program.

program	size
arr_search_static	607K
arr_search_dynamic	14K

Table 1. The list of the size of each program.

The statically compiled program is much larger than the dynamically compiled program because the statically compiled program includes all the library code in the executable file, while the dynamically compiled program places calls to the code of shared libraries during runtime.

3.

A terminal window with a dark purple background and light green text. The window title is 'yihua@ECE4730J-FA2021-Group3: ~/test_programs'. The terminal shows two commands being executed: 'time ./arr_search_dynamic 100000' and 'time ./arr_search_static 100000'. Each command is followed by its output, which includes the number of iterations completed and a breakdown of execution time into real, user, and system components.

```
yihua@ECE4730J-FA2021-Group3: ~/test_programs
yihua@ECE4730J-FA2021-Group3:~/test_programs$ time ./arr_search_dynamic 100000
./arr_search_dynamic completed 100000 iterations

real    0m1.042s
user    0m1.036s
sys     0m0.005s
yihua@ECE4730J-FA2021-Group3:~/test_programs$ time ./arr_search_static 100000
./arr_search_static completed 100000 iterations

real    0m1.033s
user    0m1.028s
sys     0m0.005s
yihua@ECE4730J-FA2021-Group3:~/test_programs$
```

Figure 3. time command.

The number of iterations used: 100000.

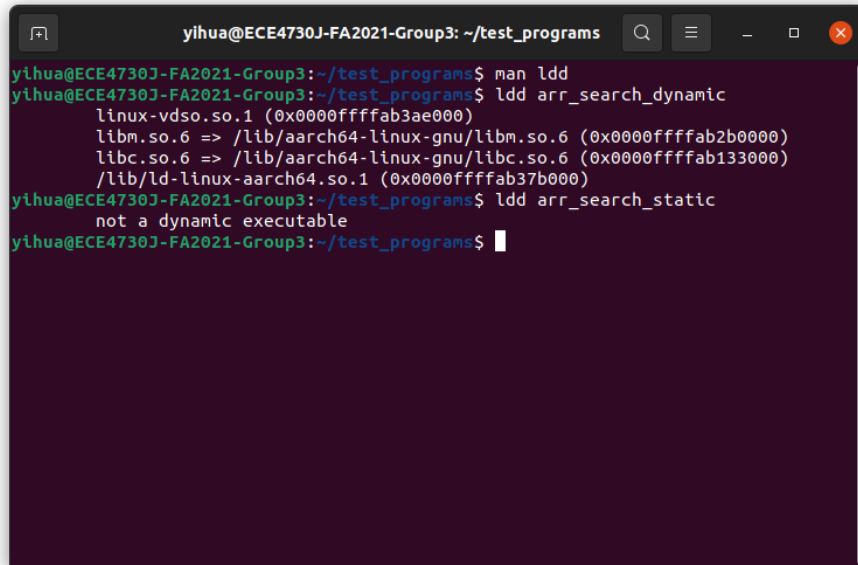
Note that there should be enough time interval between two measures of time, otherwise the measurement will be inaccurate.

The "real" execution time for the arr_search_dynamic binary: 1.042s.

The "real" execution time for the arr_search_static binary: 1.033s

The statically compiled program is slightly faster than the dynamically compiled program because the statically compiled program can directly execute the library code without calling to shared library. However, the difference is very small. If you repeat many times, you will find that sometimes they almost have the same execution time.

4.

A terminal window with a dark background and light-colored text. The window title is 'yihua@ECE4730J-FA2021-Group3: ~/test_programs'. The user enters 'man ldd' and then 'ldd arr_search_dynamic'. The output shows the shared libraries for 'arr_search_dynamic': 'linux-vdso.so.1 (0x0000ffffab3ae000)', 'libm.so.6 => /lib/aarch64-linux-gnu/libm.so.6 (0x0000ffffab2b0000)', 'libc.so.6 => /lib/aarch64-linux-gnu/libc.so.6 (0x0000ffffab133000)', and '/lib/ld-linux-aarch64.so.1 (0x0000ffffab37b000)'. Then the user enters 'ldd arr_search_static' and the output is 'not a dynamic executable'.

```
yihua@ECE4730J-FA2021-Group3:~/test_programs$ man ldd
yihua@ECE4730J-FA2021-Group3:~/test_programs$ ldd arr_search_dynamic
linux-vdso.so.1 (0x0000ffffab3ae000)
libm.so.6 => /lib/aarch64-linux-gnu/libm.so.6 (0x0000ffffab2b0000)
libc.so.6 => /lib/aarch64-linux-gnu/libc.so.6 (0x0000ffffab133000)
/lib/ld-linux-aarch64.so.1 (0x0000ffffab37b000)
yihua@ECE4730J-FA2021-Group3:~/test_programs$ ldd arr_search_static
not a dynamic executable
yihua@ECE4730J-FA2021-Group3:~/test_programs$
```

Figure 4. ldd command.

They are different because `ldd` is used to print shared object dependencies. The statically compiled program does not require shared objects (shared libraries), so it prompts "not a dynamic executable". On the contrary, the dynamically compiled program requires shared objects (shared libraries), so `ldd` prints the shared objects (shared libraries) it used:

```
linux-vdso.so.1
libm.so.6 => /lib/aarch64-linux-gnu/libm.so.6
libc.so.6 => /lib/aarch64-linux-gnu/libc.so.6
/lib/ld-linux-aarch64.so.1
```

5.

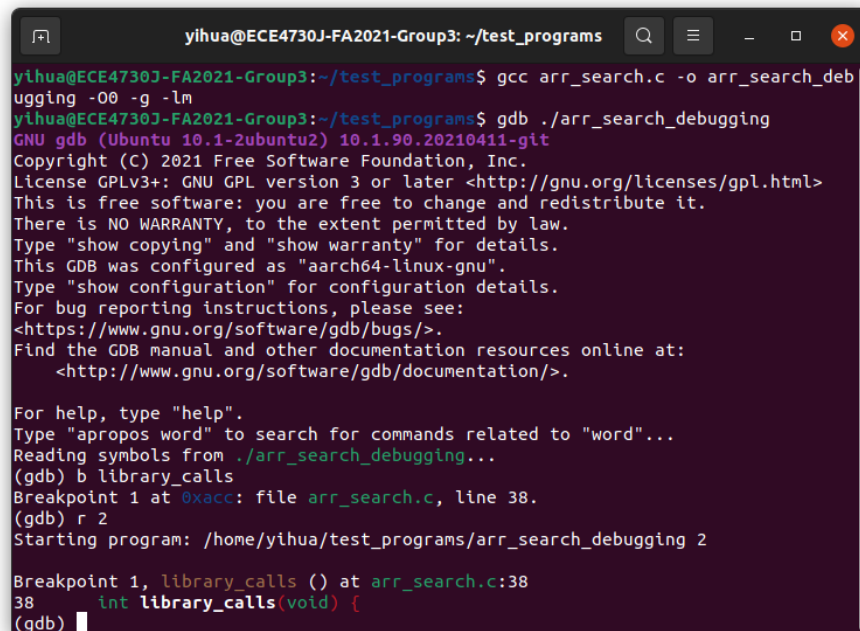
A terminal window with a dark background and light-colored text. The window title is 'yihua@ECE4730J-FA2021-Group3: ~/test_programs'. The user enters the command 'gcc arr_search.c -o arr_search_debugging -O0 -g -lm'. The prompt changes to 'yihua@ECE4730J-FA2021-Group3:~/test_programs\$'. The user then enters 'gdb ./arr_search_debugging'. The terminal displays the GNU gdb version 10.1.90.20210411-git, copyright information for the Free Software Foundation, Inc., and the GNU GPL license. It also shows the configuration as 'aarch64-linux-gnu'. The user enters 'b library_calls', and the terminal shows 'Breakpoint 1 at 0xacc: file arr_search.c, line 38.'. The user enters 'r 2', and the terminal shows 'Starting program: /home/yihua/test_programs/arr_search_debugging 2'. The terminal then shows 'Breakpoint 1, library_calls () at arr_search.c:38' and the start of the function definition '38 int library_calls(void) {'.

Figure 7. Compiling and loading programs by gdb.

Using `-g` option to compile our program by `gcc` we can use `gdb` to debug.

7. `b` command adds a breakpoint, where `library_calls` is the function. `r` commands runs the program, and 2 is the number of iterations.

8.

```

Starting program: /home/yihua/test_programs/arr_search_debugging 2

Breakpoint 1, library_calls () at arr_search.c:38
38     int library_calls(void) {
(gdb) c
Continuing.

Breakpoint 1, library_calls () at arr_search.c:38
38     int library_calls(void) {
(gdb) n
45         values = (float *) malloc(ARR_SIZE * sizeof(float));
(gdb) p values
$1 = (float *) 0xaaaaaaab32a0
(gdb) watch *(float *) 0xaaaaaaab32a0
Hardware watchpoint 2: *(float *) 0xaaaaaaab32a0
(gdb) watch *(float *) (values + 1)
Hardware watchpoint 3: *(float *) (values + 1)
(gdb) c
Continuing.

Hardware watchpoint 2: *(float *) 0xaaaaaaab32a0

Old value = 0
New value = 1
library_calls () at arr_search.c:49
49     for (i=0; i<ARR_SIZE; i++) {
(gdb)

```

Figure 8. Executing the program in gdb.

c command continues to the next breakpoint, and we can see that the program stops again at the function `library_calls`, which indicates that we are executing the second time.

9. We can see that the address of the variable `values` is `0xaaaaaaab32a0`.
- 10.

```

library_calls () at arr_search.c:49
49     for (i=0; i<ARR_SIZE; i++) {
(gdb) c
Continuing.

Hardware watchpoint 3: *(float *) (values + 1)

Old value = 0
New value = 1.41421354
library_calls () at arr_search.c:49
49     for (i=0; i<ARR_SIZE; i++) {
(gdb) c
Continuing.

Hardware watchpoint 2: *(float *) 0xaaaaaaab32a0

Old value = 1
New value = 0
memset_s (0) at /usr/include/stdlib.h:587
587     __sysdep/arch4/memset.S: No such file or directory.
(gdb) c
Continuing.

Watchpoint 3 deleted because the program has left the block in
which its expression is valid.
main (argc=2, argv=0xfffff06528) at arr_search.c:88
88     return library_calls();
(gdb) c
Continuing.

Hardware watchpoint 2: *(float *) 0xaaaaaaab32a0

Old value = 0
New value = 6.58610278e-44
__new_file_operations (&0xfffff06528->_IO_2_1_stdout_, ch=47) at fileops.c:782
782     fileops.c: No such file or directory.
(gdb) c
Continuing.

Hardware watchpoint 2: *(float *) 0xaaaaaaab32a0

Old value = 6.58610278e-44
New value = 3.73740313e-41
__new_file_operations (&0xfffff06528->_IO_2_1_stdout_, ch=104) at fileops.c:782
782     in fileops.c
(gdb) c
Continuing.

Hardware watchpoint 2: *(float *) 0xaaaaaaab32a0

Old value = 3.73740313e-41
New value = 1.0231154e-38
__new_file_operations (&0xfffff06528->_IO_2_1_stdout_, ch=111) at fileops.c:782
782     in fileops.c
(gdb)

```

Figure 9. Continue execution until the program execution exits.

```
yihua@ECE4730J-FA2021-Group3: ~/test_programs
782
782      in fileops.c
(gdb) c
Continuing.

Hardware watchpoint 2: *(float *) 0xaaaaaab32a0

Old value = 3.73740313e-41
New value = 1.02311141e-38
_IO_new_file_overflow (f=0xffffffff7f06520 <_IO_2_1_stdout_>, ch=111) at fileops.c:
782
782      in fileops.c
(gdb) c
Continuing.

Hardware watchpoint 2: *(float *) 0xaaaaaab32a0

Old value = 1.02311141e-38
New value = 4.63080422e+27
_IO_new_file_overflow (f=0xffffffff7f06520 <_IO_2_1_stdout_>, ch=109) at fileops.c:
782
782      in fileops.c
(gdb) c
Continuing.
/home/yihua/test_programs/arr_search_debugging completed 2 iterations
[Inferior 1 (process 4370) exited normally]
(gdb)
```

Figure 10. Continue execution until the program execution exits.

Values of values[0]: 0 - > 1 - > 0 - > 6.58610278e - 44 - >
3.73740313e - 41 - > 1.02311141e - 38 - > 4.63080422e + 27
Values of values[1]: 0 - > 1.41421354