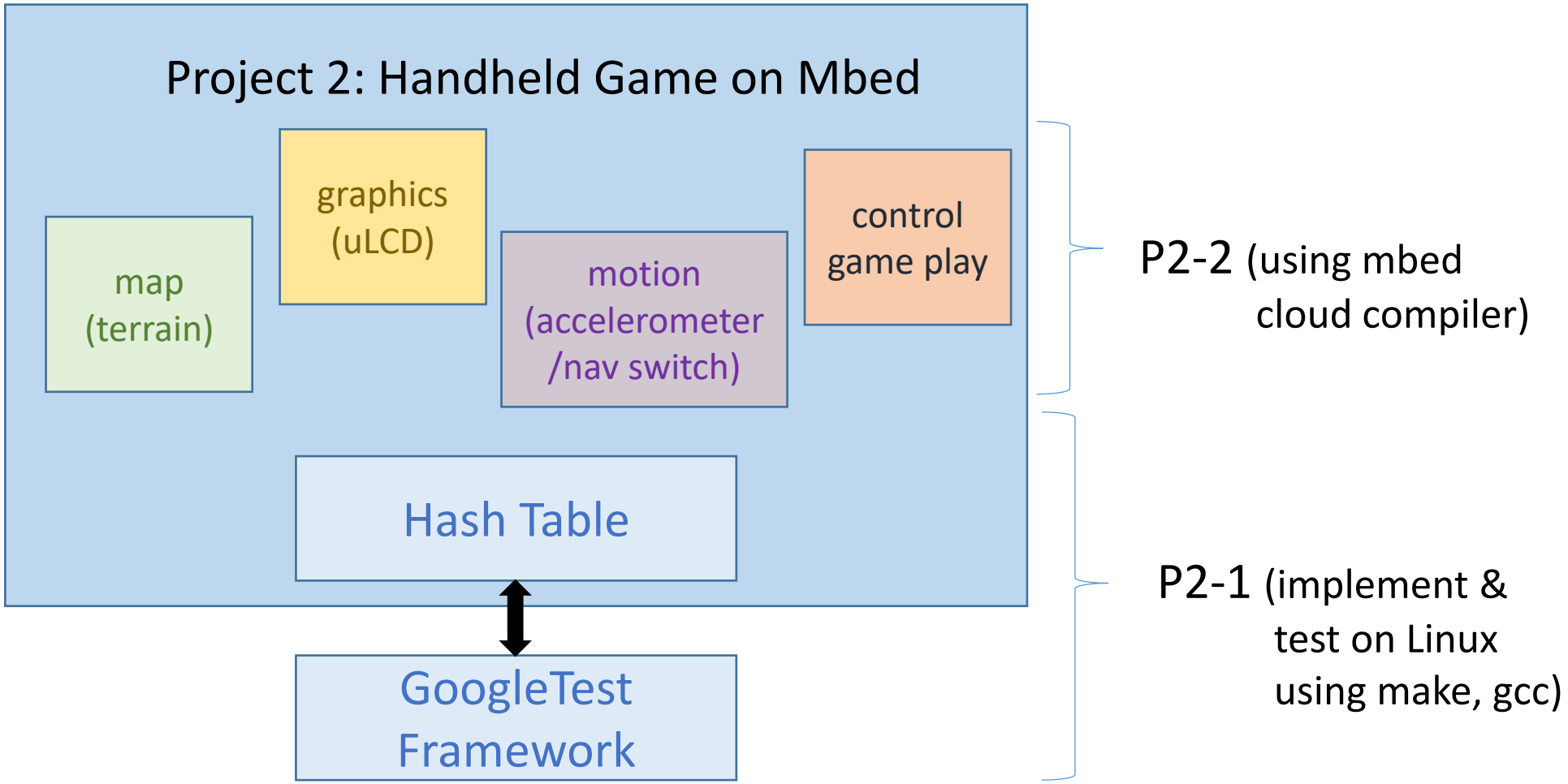
Project 2-1: Hash Table Implementation and Test



Useful Tools

1. **make**: a build tool to ease compilation and run code
2. **Google Test Framework**: a library for writing and automatically executing tests;
3. **Address Sanitizer**: a Linux-based tool for detecting memory errors.

1. Makefile

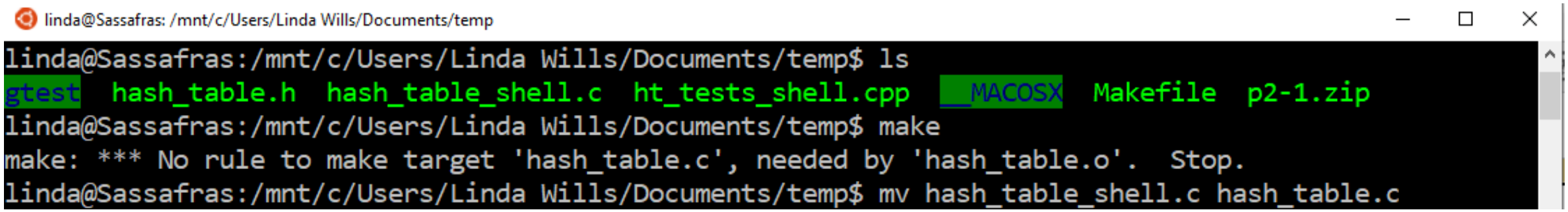
- multi-file compilation
- specifies dependencies between files
- makes it easier to re-compile correct files when change is made
- like a script – simple “make” command rather than

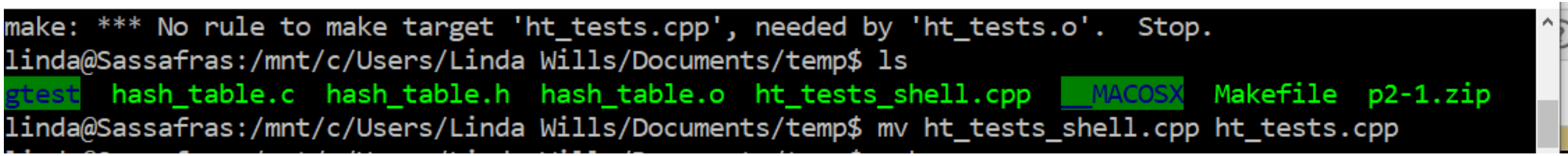
```
> gcc -g -Wall -… file0.c -o...
> gcc -g -Wall -… file1.c -o myExecutable
> ./myExecutable arg1 arg2…
```

just type:

```
> make
```

Don’t forget to rename “shell” files





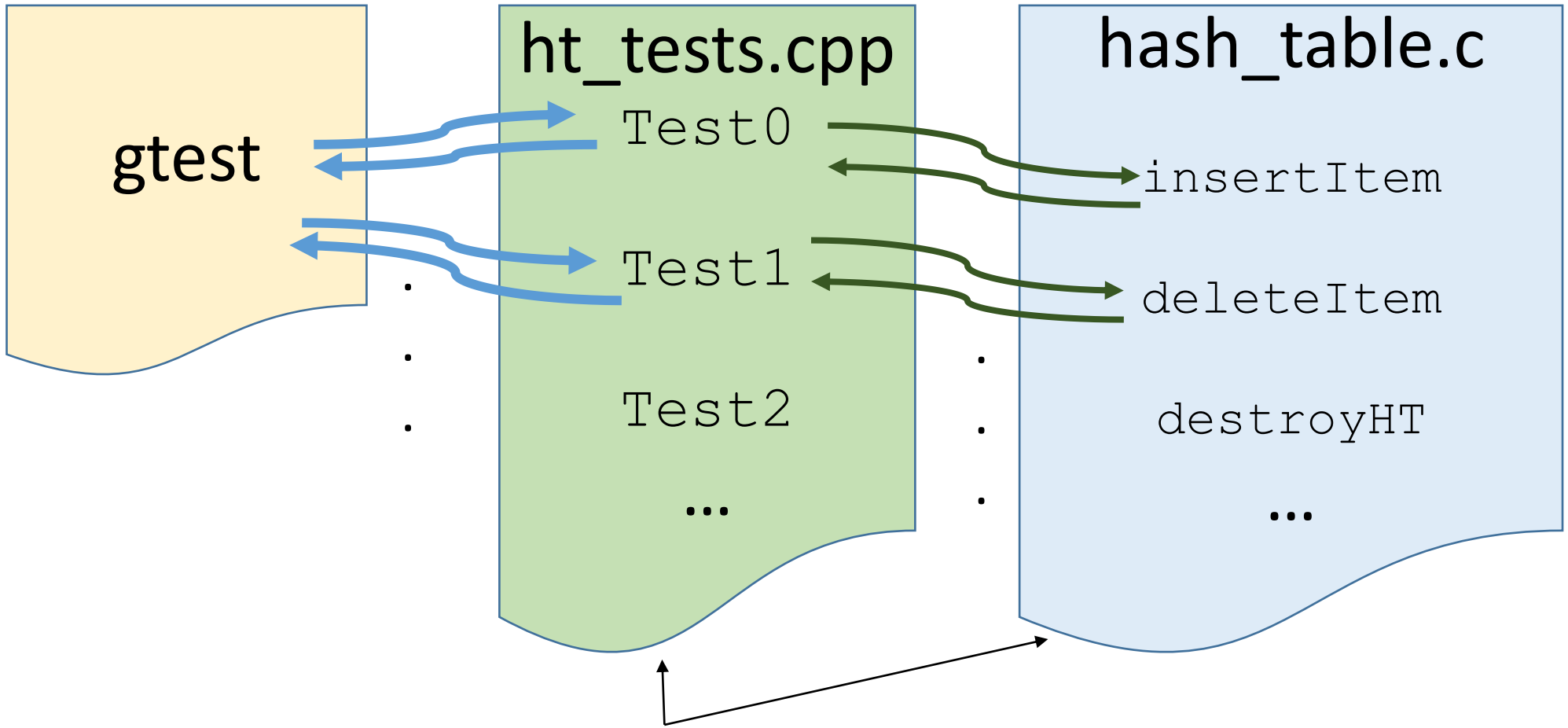
When run make, if you get “pthread” errors…

```
linda@Sassafras:/mnt/c/Users/Linda Wills/Documents/temp$ make
g++ -isystem gtest/include -g -Wall -Wextra  -c ht_tests.cpp
g++ -isystem gtest/include -Igtest -g -Wall -Wextra  -c \
            gtest/src/gtest-all.cc
g++ -isystem gtest/include -Igtest -g -Wall -Wextra  -c \
            gtest/src/gtest_main.cc
ar rv gtest_main.a gtest-all.o gtest_main.o
ar: creating gtest_main.a
a - gtest-all.o
a - gtest_main.o
g++ -isystem gtest/include -g -Wall -Wextra  -lpthread hash_table.o ht_tests.o gtest_main.a -o ht_te
sts
gtest_main.a(gtest-all.o): In function `testing::internal::ThreadLocal<testing::TestPartResultReport
erInterface*>::~ThreadLocal()':
/mnt/c/Users/Linda Wills/Documents/temp/gtest/include/gtest/internal/gtest-port.h:2050: undefined re
ference to `pthread_getspecific'
/mnt/c/Users/Linda Wills/Documents/temp/gtest/include/gtest/internal/gtest-port.h:2054: undefined re
ference to `pthread_key_delete'
gtest_main.a(gtest-all.o): In function `testing::internal::ThreadLocal<std::vector<testing::internal
::TraceInfo, std::allocator<testing::internal::TraceInfo> > >::~ThreadLocal()':
/mnt/c/Users/Linda Wills/Documents/temp/gtest/include/gtest/internal/gtest-port.h:2050: undefined re
ference to `pthread_getspecific'
/mnt/c/Users/Linda Wills/Documents/temp/gtest/include/gtest/internal/gtest-port.h:2054: undefined re
```

In Makefile, uncomment -pthread flag:

CXXFLAGS += -g -Wall -Wextra #-pthread

2. Unit Testing w/ Google Test Framework (GTest)



You write these (ht_test_shell.cpp and hash_table_shell.c get you started).

Sample Tests

Group

Test name

Expectations test will verify

Creating dummy data values to insert into ht

```
////////////////
// Access tests
////////////////
TEST(AccessTest, GetKey_TableEmpty)
{
        HashTable* ht = createHashTable(hash, BUCKET_NUM);

        // Test when table is empty.
        EXPECT_EQ(NULL, getItem(ht, 0));
        EXPECT_EQ(NULL, getItem(ht, 1));
        EXPECT_EQ(NULL, getItem(ht, 2));

        // Test with index greater than the number of buckets.
        EXPECT_EQ(NULL, getItem(ht, 10));

        destroyHashTable(ht);
}

TEST(AccessTest, GetSingleKey)
{
  HashTable* ht = createHashTable(hash, BUCKET_NUM);

  // Create list of items
  size_t num_items = 1;
  HTItem* m[num_items];
  make_items(m, num_items);

  insertItem(ht, 0, m[0]);
  EXPECT_EQ(m[0], getItem(ht, 0));

  destroyHashTable(ht);    // dummy item is also freed here
}
```

Unit Testing Method

- Incremental Design and Test
  - Write Test before Code that satisfies it
  - Test gives expectation for output
  - Run gtest: should fail Test
  - Write Code to make Test succeed
  - Write new Test2 that fails
  - Write Code to make Test2 succeed
  - … repeat

```
Running main() from gtest_main.cc
[==========] Running 7 tests from 4 test cases.
[----------] Global test environment set-up.
[----------] 1 test from InitTest
[ RUN      ] InitTest.CreateDestroyHashTable
[       OK ] InitTest.CreateDestroyHashTable (0 ms)
[----------] 1 test from InitTest (3 ms total)

[----------] 3 tests from AccessTest
[ RUN      ] AccessTest.GetKey_TableEmpty
```

```
      Expected: m[0]
      Which is: 0x2480270
To be equal to: insertItem(ht, 0, m[1])
      Which is: 0x247f7f0
ht_tests.cpp:184: Failure
      Expected: m[1]
      Which is: 0x247f590
To be equal to: getItem(ht,0)
      Which is: 0x247f7f0
[  FAILED  ] InsertTest.InsertAsOverwrite (8 ms)
[----------] 1 test from InsertTest (9 ms total)

[----------] Global test environment tear-down
[==========] 7 tests from 4 test cases ran. (105 ms total)
[  PASSED  ] 1 test.
[  FAILED  ] 6 tests, listed below:
[  FAILED  ] AccessTest.GetKey_TableEmpty
[  FAILED  ] AccessTest.GetSingleKey
[  FAILED  ] AccessTest.GetKey_KeyNotPresent
[  FAILED  ] RemoveTest.SingleValidRemove
[  FAILED  ] RemoveTest.SingleInvalidRemove
[  FAILED  ] InsertTest.InsertAsOverwrite

 6 FAILED TESTS
Makefile:19: recipe for target 'test' failed
make: *** [test] Error 1
linda@Sassafras:/mnt/c/Users/Linda Wills/Documents/temp$
```

Gtest reruns all tests after each modification to code to see if code change caused any to fail.

3. Detecting Memory Access Errors (**Address Sanitizer**)

```
> make leak
```

Runs w/ gcc.
If it's not installed and configured on your local machine, log in to remote ECE Linux server and run it there (see tutorial).

```
linda@LILAC: /mnt/c/Users/Linda Wills/Documents/Assignments/P2-1/addr-sanitizer-tutorial
linda@LILAC:/mnt/c/Users/Linda Wills/Documents/Assignments/P2-1/addr-sanitizer-tutorial$ make clean
rm dll dll.o
linda@LILAC:/mnt/c/Users/Linda Wills/Documents/Assignments/P2-1/addr-sanitizer-tutorial$ make leak
cc -g -Wall -std=c11 -fsanitize=address -ggdb  -c dll.c
cc -g -Wall -std=c11 -fsanitize=address -ggdb  -o dll dll.o
linda@LILAC:/mnt/c/Users/Linda Wills/Documents/Assignments/P2-1/addr-sanitizer-tutorial$ ./dll

=================================================================
==1008==ERROR: LeakSanitizer: detected memory leaks

Direct leak of 16 byte(s) in 1 object(s) allocated from:
    #0 0x7f5769633808 in __interceptor_malloc ../../../../src/libsanitizer/asan/asan_malloc_linux.cc:144
    #1 0x7f5769f98409 in insert /mnt/c/Users/Linda Wills/Documents/Assignments/P2-1/addr-sanitizer-tutorial/dll.c:46
    #2 0x7f5769f98520 in main /mnt/c/Users/Linda Wills/Documents/Assignments/P2-1/addr-sanitizer-tutorial/dll.c:61
    #3 0x7f57693540b2 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x240b2)

Direct leak of 16 byte(s) in 1 object(s) allocated from:
    #0 0x7f5769633808 in __interceptor_malloc ../../../../src/libsanitizer/asan/asan_malloc_linux.cc:144
    #1 0x7f5769f98409 in insert /mnt/c/Users/Linda Wills/Documents/Assignments/P2-1/addr-sanitizer-tutorial/dll.c:46
    #2 0x7f5769f98536 in main /mnt/c/Users/Linda Wills/Documents/Assignments/P2-1/addr-sanitizer-tutorial/dll.c:62
    #3 0x7f57693540b2 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x240b2)

Direct leak of 16 byte(s) in 1 object(s) allocated from:
    #0 0x7f5769633808 in __interceptor_malloc ../../../../src/libsanitizer/asan/asan_malloc_linux.cc:144
    #1 0x7f5769f98409 in insert /mnt/c/Users/Linda Wills/Documents/Assignments/P2-1/addr-sanitizer-tutorial/dll.c:46
    #2 0x7f5769f9850a in main /mnt/c/Users/Linda Wills/Documents/Assignments/P2-1/addr-sanitizer-tutorial/dll.c:60
    #3 0x7f57693540b2 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x240b2)

SUMMARY: AddressSanitizer: 48 byte(s) leaked in 3 allocation(s).
linda@LILAC:/mnt/c/Users/Linda Wills/Documents/Assignments/P2-1/addr-sanitizer-tutorial$
```

Tutorial on Address Sanitizer: Mem-error-detection-w-Addr-Sanitizer.zip (linked from on Canvas P2-1 Assignment page)

Goal

- Complete hash_table.c (refer to hash_table.h for specification)
- Add tests to ht_tests.cpp to fully exercise your code
- Full credit if your hash_table code:
  - Is correct (passes all of our tests)
  - Has no memory access errors
  - Compiles (builds with make) w/out errors/warnings

- Submit zip of files:
  - **`hash_table.h`**
  - **`hash_table.c`**
  - **`ht_tests.cpp`**

Important Files and Documentation

P2-1 HashTable Implementation and Test.pdf – read this carefully

p2-1.zip:

1. ht_tests_shell.cpp provides some tests; you need to add more.
2. hash_table_shell.c contains some code that passes some of the given tests, but you need to add code to make the other tests succeed.
3. hash_table.h provides documentation on functions you need to write

Hints to guide you thru this process are in

P2-1-incremental_design_and_test.pdf

Tutorial on gtest and make: gtest_hashtable.pdf

Tutorial on Address Sanitizer: Mem-error-detection-w-Addr-Sanitizer.zip