Embedded Systems Programming
Assignment 4.1
Embedded Linux

Steinarr Hrafn Höskuldsson

September 28, 2022

Part 1

Using Rpi-Imager, a SD card was formatted and the Raspberry Pi was then booted. A terminal window was used to confirm that one could ssh onto it. SSH FS extension for VSCode was installed and after configuring SSH FS, the SSH client could connect to the Raspberry Pi. A workspace was created and a simple program was written that prints "Hello" to stdout.

```
#include <stdio.h>

int main()
{
    printf("Hello\n");
    return 0;
}
```

Listing 1: src/hello.cpp, writes "Hello" to stdout

It compiled and ran as expected. Next a makefile was created:

```
# This is the default target, which will be built when
# you invoke make
.PHONY: all
all: hello
# This rule tells make how to build the file hello from hello.cpp
hello: src/hello.cpp
   g++ -o hello src/hello.cpp
# This rule tells make to copy hello to the binaries subdirectory,
# creating it if necessary
.PHONY: install
install:
   mkdir -p bin
   cp -p hello bin
# This rule tells make to delete hello
.PHONY: clean
clean:
   rm -f hello
```

Listing 2: Makefile for simple hello program

Part 2

The Mutex example was copied and split into header files in `include`, implementation files in `src` and `src/main_count.cpp`. They can be seen in Appendix A. A Makefile was created and configured such that it will only compile the changed implementation files.

```
BUILD=build
# This is the default target, which will be built when
# you invoke make
.PHONY: all
all: main_count
# This rule tells make how to build the file hello from hello.cpp

hello: src/hello.cpp
   g++ -o hello src/hello.cpp

# build source files into objects files
$(BUILD)/%.o : src/%.cpp
   g++ -pthread -o $@ -c $< -I include

main_count: $(BUILD)/main_count.o $(BUILD)/decrement.o $(BUILD)/increment.o
   g++ -pthread -o $@ $^
#  g++ -pthread -o main src/main.cpp -I include

# This rule tells make to copy hello to the binaries subdirectory,
# creating it if necessary
.PHONY: install
install:
   mkdir -p bin
   cp -p hello bin
# This rule tells make to delete hello
.PHONY: clean
clean:
   rm -f hello
   rm -f $(BUILD)/*.o
   rm -f main
```

Listing 3: Makefile for compiling Mutex example

Part 3

The `increment` and `decrement` implementations were copied to `increment_fifo` and `decrement_fifo` and modified so that increment adds an integer to the queue and `decrement` gets an integer, the implementations can be seen in Appendix B. The `Fifo` implementation was modified so that the inner workings of it were protected with the MutEx.

```
#include "fifo3.h"

Fifo::Fifo()
{
    reset();
}

int* Fifo::incr_p(int* p){
    // this function increments the pointers in the ring buffer
    if (p + 1 == &buffer[0] + FIFO_SIZE)

    // if we're about to exit the buffer
        return &buffer[0]; // return to 0
    else
        return p+1;
```

```
}

int Fifo::get()
{
    // get shouldn't be called on an empty queue but:
    if (is_empty())
    {
        return -1;
    }

    int ret = *head; // get first item
    head = incr_p(head); // move the head by 1

    // if the queue is now empty:
    if (head == tail) it_is_empty = true;

    return ret;
}

void Fifo::put(int item)
{
    if (is_full())
    {
        get();  // throw one out to make room
        put(item); //recursive for the win!
        return;
    }
    *tail = item; // put uten into queue
    tail = incr_p(tail); // increment tail
    it_is_empty = false; // no longer empty
}

bool Fifo::is_empty()
{
    return it_is_empty;
}

bool Fifo::is_full()
{
    return (head == tail && !it_is_empty);
}

void Fifo::reset()
{
    tail = &buffer[0];
    head = &buffer[0];
    it_is_empty = true;
}
```

Listing 4: Thread safe implementation of a First-in-First-out Queue.

`main_fifo.cpp` was written to start the `increment` and `decrement` threads.

```
#include <pthread.h>
#include <stdio.h>
#include <unistd.h>
#include <stdint.h>
#include "fifo3.h"
#include "increment_fifo.h"
#include "decrement_fifo.h"

pthread_mutex_t sharedVariableMutex;
```

```
Fifo fifo;

pthread_t incrementTaskObj;
pthread_t decrementTaskObj;


int main(void)
{
    /* Create the mutex for accessing the shared variable using the
    * default attributes. */
    pthread_mutex_init(&sharedVariableMutex, NULL);
    /* Create the increment and decrement tasks using the default task
    * attributes. Do not pass in any parameters to the tasks. */
    pthread_create(&incrementTaskObj, NULL, incrementTask, NULL);
    pthread_create(&decrementTaskObj, NULL, decrementTask, NULL);
    /* Allow the tasks to run. */
    pthread_join(incrementTaskObj, NULL);
    pthread_join(decrementTaskObj, NULL);
return 0;
}
```

Listing 5: src/main_fifo.cpp

The Makefile was edited to compile `main_fifo`

```
BUILD=build
# This is the default target, which will be built when
# you invoke make
.PHONY: all
all: main_fifo
# This rule tells make how to build the file hello from hello.cpp

hello: src/hello.cpp
   g++ -o hello src/hello.cpp

# build source files into objects files
$(BUILD)/%.o : src/%.cpp
   g++ -pthread -o $@ -c $< -I include

main_count: $(BUILD)/main_count.o $(BUILD)/decrement.o $(BUILD)/increment.o
   g++ -pthread -o $@ $^
#  g++ -pthread -o main src/main.cpp -I include

main_fifo: $(BUILD)/main_fifo.o $(BUILD)/decrement_fifo.o $(BUILD)/increment_fifo
    .o
   g++ -pthread -o $@ $^
# This rule tells make to copy hello to the binaries subdirectory,
# creating it if necessary
.PHONY: install
install:
   mkdir -p bin
   cp -p hello bin
# This rule tells make to delete hello
.PHONY: clean
clean:
   rm -f hello
   rm -f $(BUILD)/*.o
   rm -f main
```

Listing 6: The Makefile used to compile main_fifo.cpp

Appendix

A Mutex example, split into header and implementation files

```
#include <pthread.h>
#include <stdint.h>

extern pthread_mutex_t sharedVariableMutex;
extern int32_t gSharedVariable;

void *incrementTask(void *param);
```

Listing 7: include/increment.h

```
#include <pthread.h>
#include <stdint.h>

extern pthread_mutex_t sharedVariableMutex;
extern int32_t gSharedVariable;

void *decrementTask(void *param);
```

Listing 8: include/decrement.h

```
#include "increment.h"
#include <stdio.h>
#include <unistd.h>


void *incrementTask(void *param)
{
    while (1)
    {
        /* Delay for 3 seconds. */
        sleep(3);
        /* Wait for the mutex to become available. */
        pthread_mutex_lock(&sharedVariableMutex);
        gSharedVariable++;
        printf("Increment Task: shared var is %d\n",
        gSharedVariable);
        /* Release the mutex for other task to use. */
        pthread_mutex_unlock(&sharedVariableMutex);
    }
}
```

Listing 9: src/increment.cpp

```
#include "decrement.h"
#include <stdio.h>
#include <unistd.h>

void *decrementTask(void *param)
{
    while (1)
    {
        /* Delay for 7 seconds. */
        sleep(7);
        /* Wait for the mutex to become available. */
        pthread_mutex_lock(&sharedVariableMutex);
```

```
        gSharedVariable--;
        printf("Decrement Task: shared var is %d\n",
        gSharedVariable);
        /* Release the mutex. */
        pthread_mutex_unlock(&sharedVariableMutex);
    }
}
```

Listing 10: src/decrement.cpp

```
#include <pthread.h>
#include <stdio.h>
#include <unistd.h>
#include <stdint.h>

pthread_mutex_t sharedVariableMutex;
int32_t gSharedVariable = 0;
#include "decrement.h"
#include "increment.h"

pthread_t incrementTaskObj;
pthread_t decrementTaskObj;


int main(void)
{
    /* Create the mutex for accessing the shared variable using the
    * default attributes. */
    pthread_mutex_init(&sharedVariableMutex, NULL);
    /* Create the increment and decrement tasks using the default task
    * attributes. Do not pass in any parameters to the tasks. */
    pthread_create(&incrementTaskObj, NULL, incrementTask, NULL);
    pthread_create(&decrementTaskObj, NULL, decrementTask, NULL);
    /* Allow the tasks to run. */
    pthread_join(incrementTaskObj, NULL);
    pthread_join(decrementTaskObj, NULL);
return 0;
}
```

Listing 11: src/main_count.cpp

B Mutex Fifo Implementation

```
#include <pthread.h>
#include <stdint.h>
#include "fifo3.h"

extern pthread_mutex_t sharedVariableMutex;
extern Fifo fifo;



void *incrementTask(void *param);
```

Listing 12: include/increment_fifo.h

```
#include <pthread.h>
#include <stdint.h>
#include "fifo3.h"
```

```
extern pthread_mutex_t sharedVariableMutex;
extern Fifo fifo;

void *decrementTask(void *param);
```

Listing 13: include/decrement_fifo.h

```
#include "increment.h"
#include <stdio.h>
#include <unistd.h>
#include "fifo3.h"

extern Fifo fifo;

void *incrementTask(void *param)
{
    int i = 0;
    while (1)
    {
        /* Delay for 3 seconds. */
        sleep(3);
        /* Wait for the mutex to become available. */
        pthread_mutex_lock(&sharedVariableMutex);
        fifo.put(i++);
        printf("Increment Task: I just put %d into fifo\n",
        i);
        /* Release the mutex for other task to use. */
        pthread_mutex_unlock(&sharedVariableMutex);
    }
}
```

Listing 14: src/increment_fifo.cpp

```
#include "decrement.h"
#include "fifo3.h"
#include <stdio.h>
#include <unistd.h>

extern Fifo fifo;

void *decrementTask(void *param)
{
    while (1)
    {
        /* Delay for 3 seconds. */
        sleep(3);
        /* Wait for the mutex to become available. */
        pthread_mutex_lock(&sharedVariableMutex);
        if (!fifo.is_empty())
        {
            int i = fifo.get();
            printf("Decrement Task: I just got %d from fifo\n",  i);
        }
        else{
            printf("Decrement Task: couldn't get, it's empty!");
        }
        /* Release the mutex for other task to use. */
        pthread_mutex_unlock(&sharedVariableMutex);
    }
}
```

Listing 15: src/decrement_fifo.cpp