Luke Pepin – CSE 4300

Assignment 5

Released: Apr 8, 2024

Due: Apr 26, 2024 11:59PM

Explanation:

The design for join/detach/exit synchronization is primarily handled in the `thread.c` file. It utilizes a combination of condition variables and locks, which have been incorporated into the structure of the thread in `thread.h` to ensure proper synchronization. Meanwhile, `menu.c` employs the `common\_prog` function from this file to create a new thread that executes a program.

Conditions:

(1) If the join occurs before the exit, the joining thread will block until the joined thread calls `pthread\_exit()`.

(2) If the detach happens before exit, once detached, the thread's resources are automatically released upon finishing execution (calling `pthread\_exit()`). The main thread does not need to join it.

(3) If the exit occurs before the join, the system will retain certain information about the exited thread (such as its exit status) until another thread joins it.

(4) If the exit happens before detach, if a thread exits before being detached, it becomes a "zombie" thread. Its resources remain unallocated until it is detached or another thread joins it.

(5) Besides these four possibilities, other scenarios may involve handling cancellation requests (`pthread\_cancel()`), signal handling, or managing threads that terminate due to unhandled exceptions.

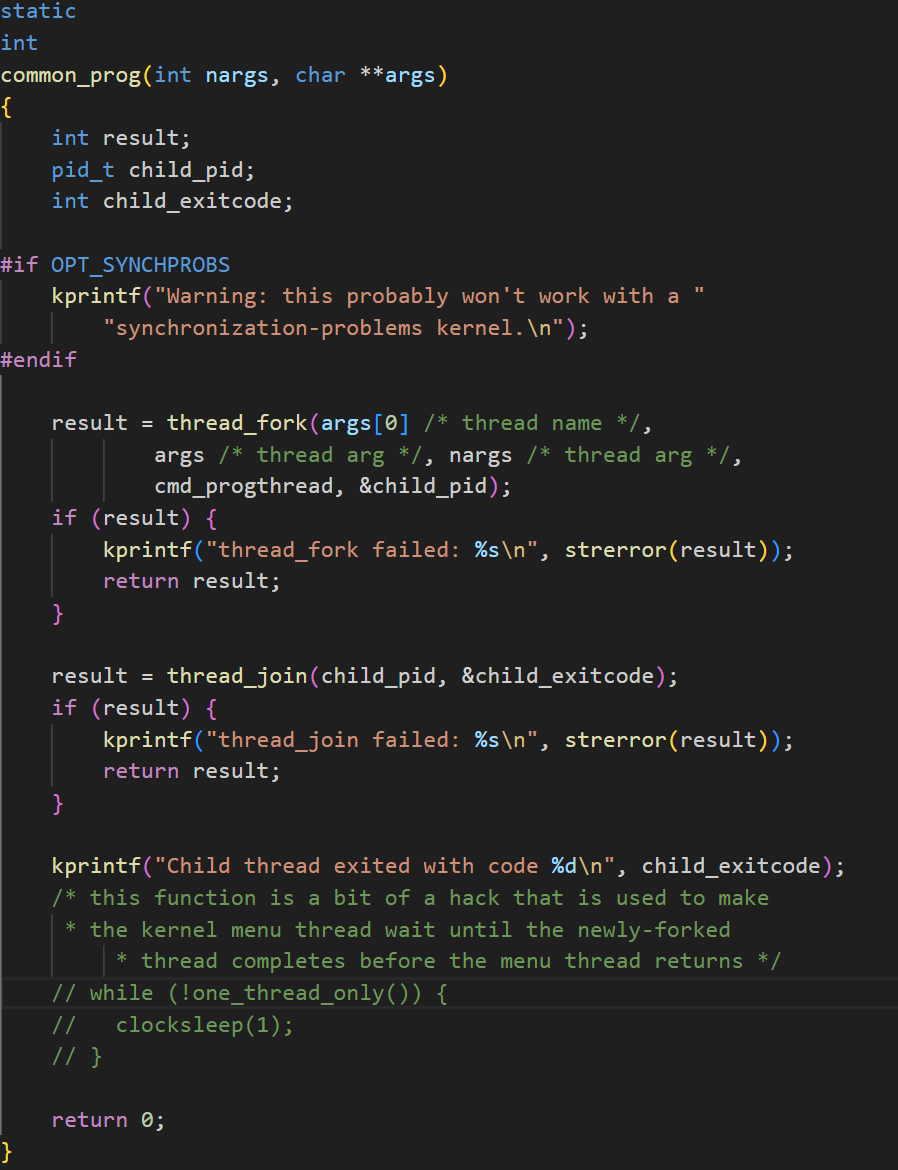
**Thread Subsystem Extension:**

menu.c

**Explanation of Changes:**

As suggested by the Assn 5 hints pptx, made the following changes to the common\_prog function, added a variable to hold `child\_pid` and the child's exit code. Used the `child\_pid` variable to store the PID from `thread\_fork`. Then, use `thread\_join`, the child's PID, and the child's `exitCode` to wait until the child thread exits before proceeding. If `thread\_join` returns an error, print it.

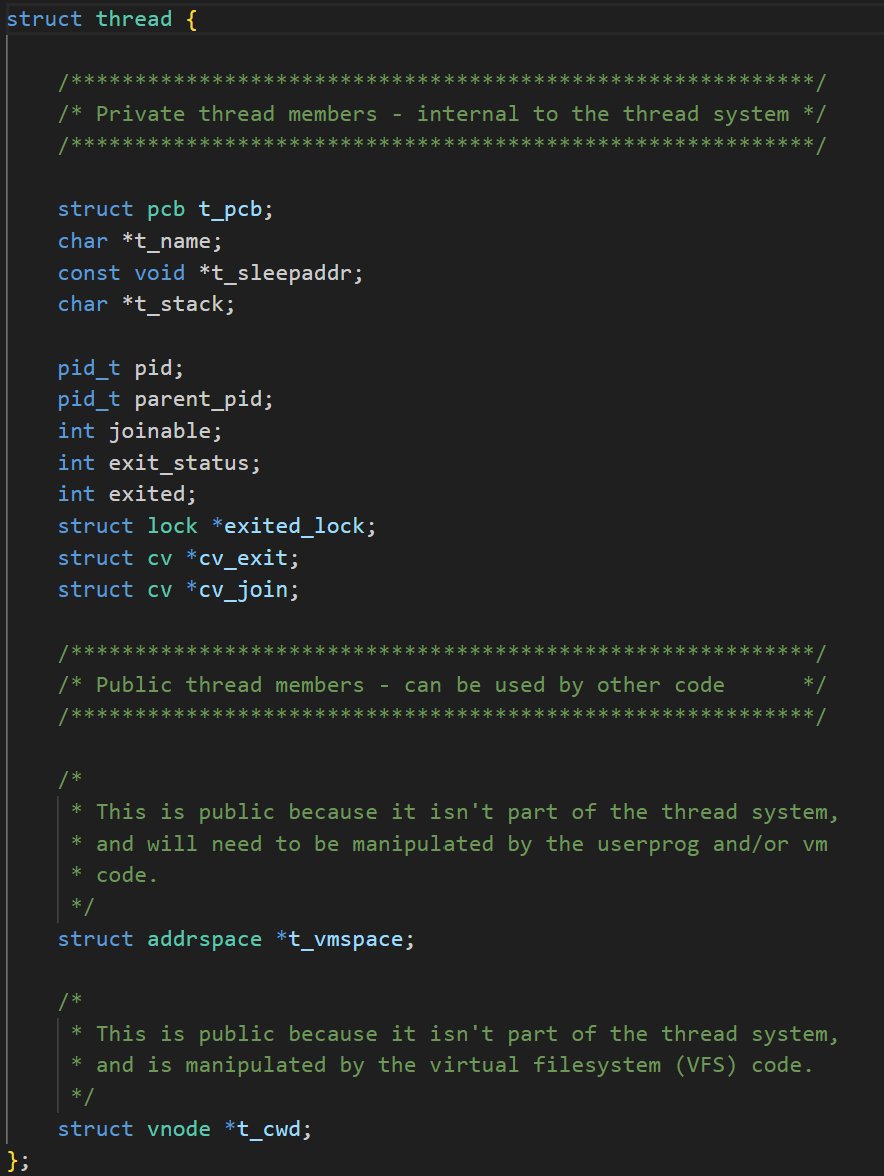
common\_prog.png

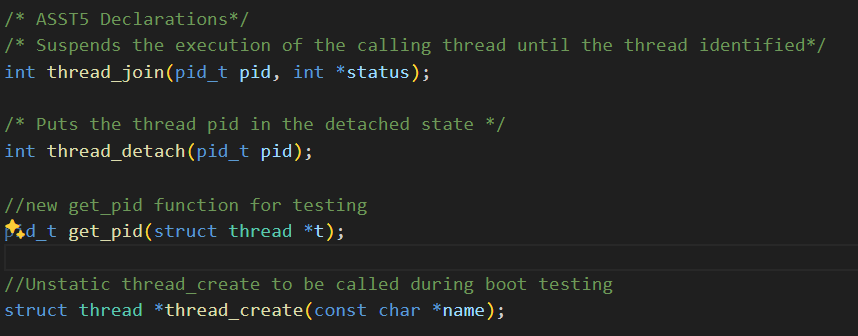


Thread.h

**Explanation of Changes:**

As suggested by the Assn 5 hints pptx, multiple new fields of the struct thread were added for the implementation of thread\_join and thread\_detach. These new fields serve a variety of purposes from tracking the pid and parent\_pid, holding status integers as well as holding lock and conditional variable structures.

thread\_structure.png 



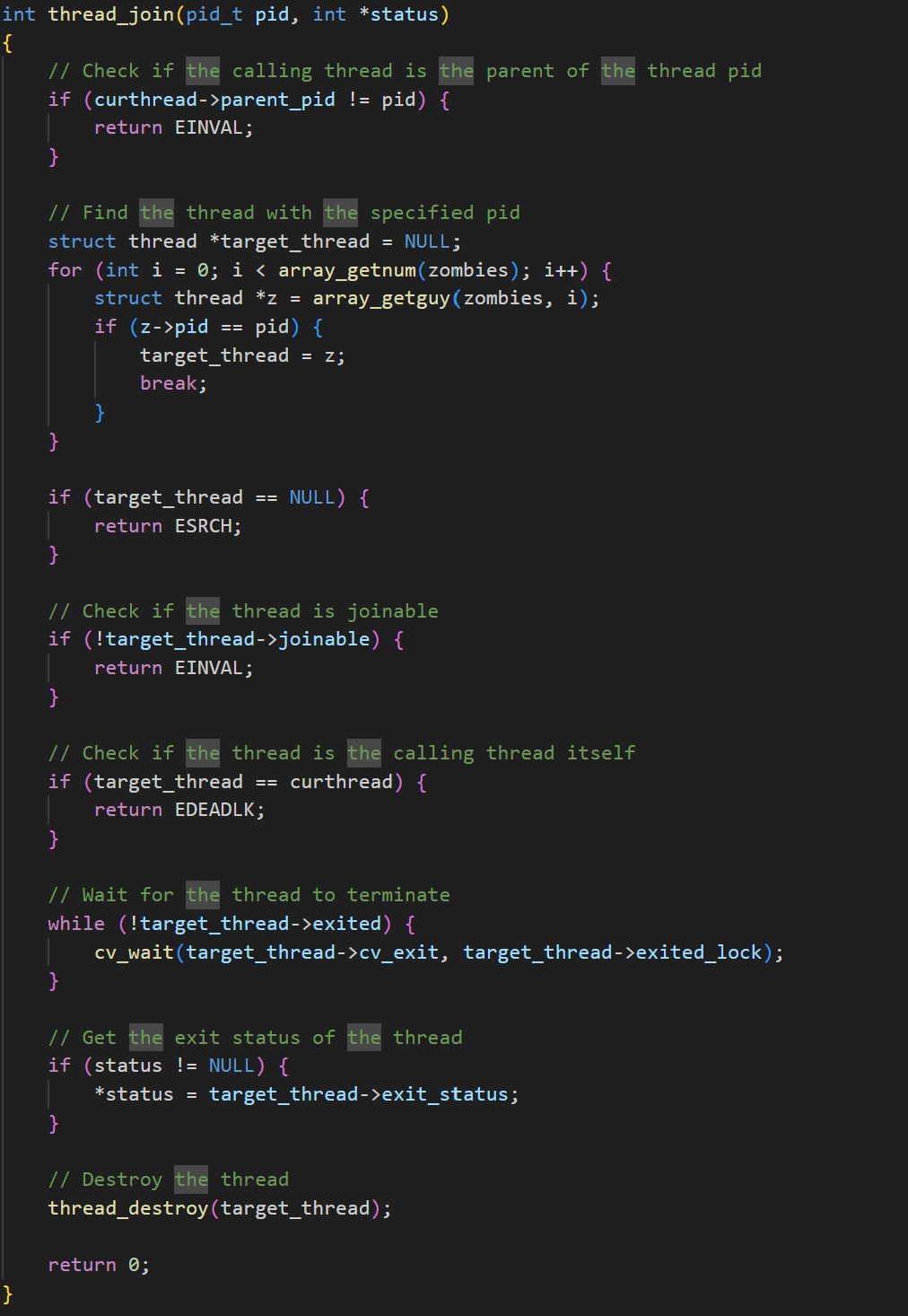
Thread.c

**Explanation of Changes thread\_join:**

As described in the assignment paper and making use of the new thread fields, thread\_join was implemented to suspend the execution of the calling thread until the thread identified by the pid terminates by calling thread exit. Under different not intended conditions different return values are adjusted.

Also #include <errno.h> and <synch.h> was a new header included

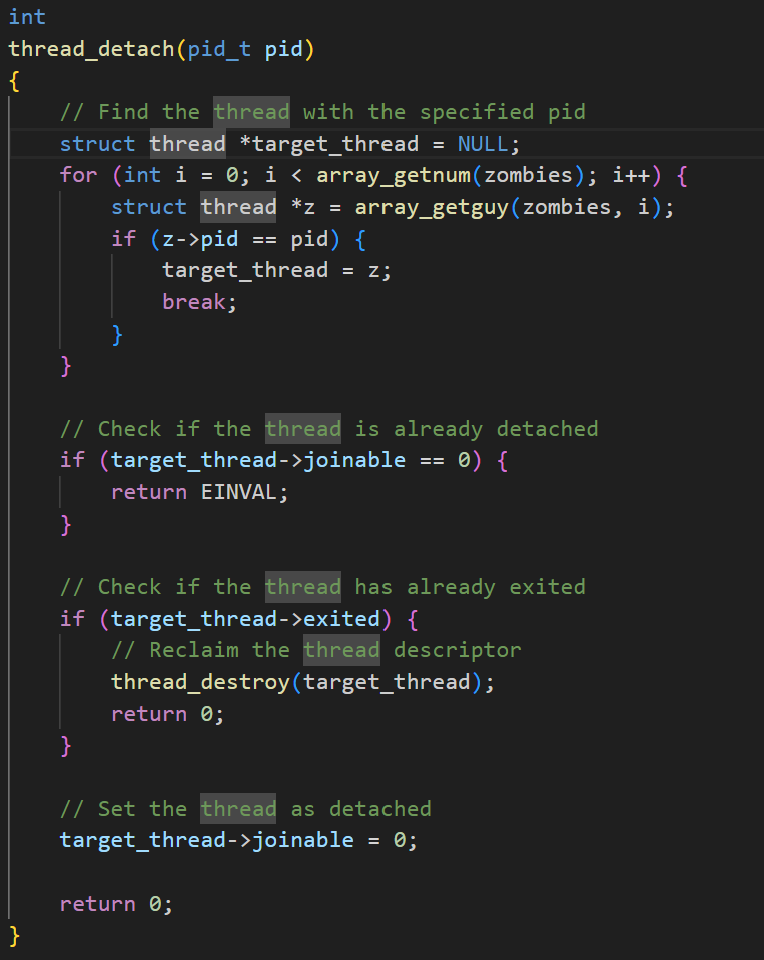
thread\_join.png



**Explanation of Changes thread\_detach:**

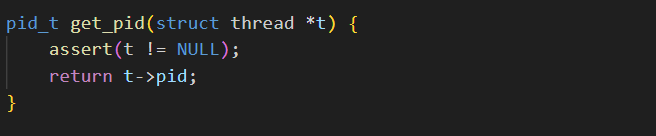
Once again as described in the assignment paper and making use of the new thread fields, thread\_detach puts the thread pid in the detached state. Meaning the thread descriptor and exit status can be discarded immediately when pid terminates.

thread\_detach.png



For testing:

Static was removed from thread\_create function & get\_pid was added, note a few imports such as #include <synch.h> and <thread.h> were added to some files to be used



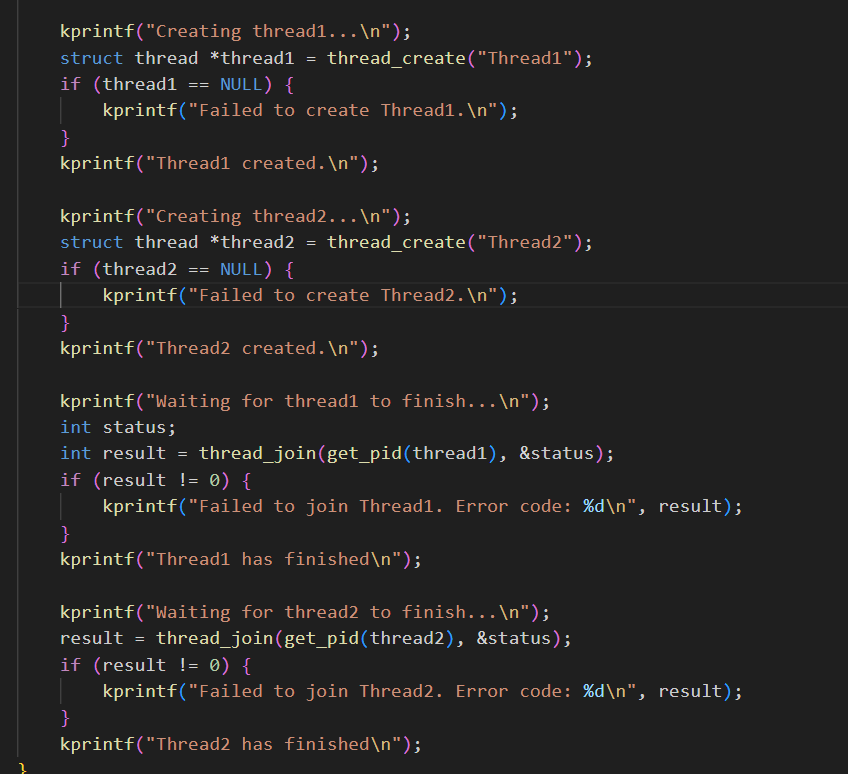
**Using Thread Join:**

Main.c

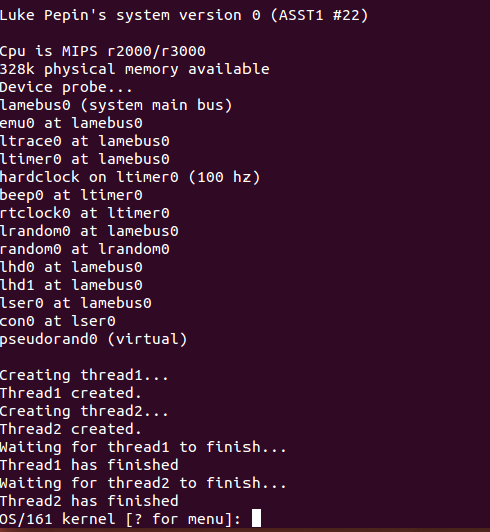
**Explanation of Change main:**

As instructed by the professor, to show the proper implementation of allowing one thread to wait for another to exit to continue, main.c calls and shows threads properly accomplishing this on startup of the kernel.

Main.png (boot(void))



menu\_thread\_wait.png



Terminal commands for ease of use:

cd

cd cs4300-os161/

cd os161-1.11 ./configure --ostree=$HOME/cs4300-os161/root --toolprefix=cs4300-

cd kern/conf

./config ASST1

cd ../compile/ASST1

make depend

make

make install

cd $HOME/cs4300-os161/os161-1.11

make

cd $HOME/cs4300-os161/root

sys161 kernel-ASST1