CSE 530 ASSIGNMENT 3 README

Files Included

1. barrier folder → contains system call source code and Makefile. Present in /linux/kernel in the patched source code.
2. Testing Program, main.c
3. Makefile that compiles main.c
4. Patch file → team12.patch to patch
5. This readme ... (of course !)

Steps

1. Download unmodified Linux source code and dot\_config file from dropbox
2. Rename dot\_config to .config
3. Copy team12.patch to parent directory of linux/
4. Open terminal and Cd into linux/
5. Apply patch using cmd patch -p1 < team12.patch
6. Make the patched kernel by following the commands given in assignment pdf
7. Boot from patched kernel
8. make main.c (test program)
9. send main(test program) to SD card root folder
10. Run ./main

Observations

There are 2 barriers per child process. First barrier has 5 threads and second barrier has 20 threads. The prints of each group of threads associated with a barrier a directed to a file. The files are

* P1\_T5.txt has prints of child process 1 barrier 1 used by 5 threads
* P1\_T20.txt has prints of child process 1 barrier 2 used by 20 threads
* P2\_T5.txt has prints of child process 2 barrier 1 used by 5 threads
* P2\_T20.txt has prints of child process 2 barrier 2 used by 20 threads

Prints common to all threads and kernel prints are printed on the console.

Test Program

Tests synchronization between 2 group of threads in each child process. First group has 5 threads and second has 20. In the first group, first 4 threads to enter barrier get blocked by the barrier. The 5th thread wakes all the 4 sleeping threads and leaves. The subsequent iteration does not start until all the threads blocked in barrier leave.

Barrier destroy

We destroy the barrier for second group of threads at the 50th iteration, but the destroy does not affect immediately. Our barrier destroy allows the threads within the barrier to finish that iteration before the barrier is destroyed. Thus a few threads that had finished try to start the next iteration and prints occur but the barrier wait call for them fails. We called pthread\_exit so threads don’t execute further iterations as it is known the barrier wait call will fail.