Pthread in xv6

Richard Hsin, Ben Werdmann, Reilly Mooney

CSSE332

Noureddine Mohammad

2/21/25

Project Overview

What is Spork

Spork is the latest and greatest addition to the xv6 operating system. Spork allows for the creation and use of lightweight threads through the user functions uspork_create and uspork_join, and their corresponding kernel functions clone and join

The thread is created in the kernel space where the clone function will take in 3 arguments. The first one being the thread function, second the argument, and third is a pre-malloced stack.

Design Decisions

The stack will be generated in the userspace function to lessen the burden of the user to malloc each time they want to create thread.

The userspace function will have an extra integer argument that can specify the size of the stack.

Stack is shared by passing the location of the allocated memory region into the clone function

Lineage

How is lineage tracked between threads?

- Lineage between threads is tracked via a linked list of process structs which maintain a link to its parents and each of its children.
- Runq helps the scheduler keep track of and choose threads for execution.

How many threads can we make?

- # Processes + # Threads (capped at NPROC)
- NPROC = 64

Memory

- Clone() calls copypasta(), which copies the memory pages of a parent process to a child process. It shares the physical pages by incrementing reference counts.

- Limitations: the caller is responsible for allocating the stack and passing it into clone as an argument.

Conclusions

One of the challenges that we faced during this project is with implementing the linked list we used to track lineage. While we were working on it we had difficulties getting the reference counter to properly track the number of threads. We were eventually able to solve this problem by creating two functions that handle the locking and counting of the reference array, inc_ref and dec_ref, inside kalloc.c.

One of the improvements that we could make in the future would be moving the allocation of the stack to the kernel space as opposed to the user space. This would save the user from having to manually allocate and hand the stack into the spork function when it is called.