SOFTENG 370 Tutorial 3

Timo van Veenendaal

18 August 2020

Hello again

- Welcome back to remote university
- Tutorials are still on at the normal time! They are recorded too!
- Want to meet? Pick a time at `https://calendly.com/timov/se370-office-hours` and I'll meet you on Zoom!
- Questions during tutorial? Unmute yourself and ask or put them in the Zoom chat
- Questions outside tutorial? Contact me, any way is good but my email is tvan508@aucklanduni.ac.nz

Plan

- Recap of `mmap`
- Condition variables
- Assignment Q&A

Example files from tutorials are available on GitHub: `https://github.com/timovv/se370-tutorials/`.

Memory mapping

- `mmap` lets you create a *memory object* (memory mapping) in the process's memory. It is a very powerful function with a lot of use cases.
- From the man page (`man mmap`):

  ```
  The mmap() function shall establish a mapping
  between an address space of a process and a
  memory object.

  The mmap() function shall be supported for
  the following memory objects:
       *  Regular files
       *  Shared memory objects
       *  Typed memory objects
  ```

Memory mapping

`mmap` has the following signature:

```
void *mmap(void *addr, size_t len, int prot, int flags,
           int fildes, off_t off);
```

- `void *addr`: address within the process's memory space where the memory mapping should be created. Pass `NULL` to let the OS to figure it out.
- `size_t len`: size (in bytes) of the memory mapping.
- `int prot`: protection flags. Some combination of `PROT_READ`, `PROT_WRITE`, and `PROT_EXEC` ORed together using `|`. Specifies what can be done with the mapped area.

Memory mapping

`mmap` has the following signature:

```
void *mmap(void *addr, size_t len, int prot, int flags,
           int fildes, off_t off);
```

- `int flags`: bitwise or of these flags: `MAP_ANONYMOUS` (mapping not attached to a file), `MAP_SHARED` (share mapping between processes) and `MAP_PRIVATE` (unique copy of memory for each process).
- `int fildes`: a file descriptor for a file previously opened using `open()`. If no file, set to -1.
- `int off`: offset into the file for the memory mapping.
- Returns a `void *` pointing to the created memory mapping.
- Q: What arguments for `mmap` would create an area for processes can communicate with?

Condition variables

- Assignment step 4
- Can be used to block a thread, or multiple threads at the same time, until another thread both modifies a shared variable (the *condition*), and signals the condition variable.
- `pthread_cond_wait`: wait on the condition variable to be signalled by another thread.
- `pthread_cond_signal`: signal the condition variable, releasing one **or more** waiting threads. Called after changing some condition.
- Used in combination with a mutex which protects the condition.
- Demo: `condition_vars.c`

Q&A

- Questions about the assignment?