ECE 3574: InterProcess Communication with Pipes and Sockets

Changwoo Min

InterProcess Communication with Pipes and Sockets

- InterProcess Communication with Pipes and SocketsToday we are going to look at concurrent programming using multiple OS processes.

    - communicating processes

    - Unix and Windows pipes

    - Unix fork + pipes

    - Cross-platform IPC using QProcess and QLocalSocket

A reoccuring theme in concurrent programming is the idea of communication over a channel.

- Depending on how these are implemented they go by many names:

  - Pipes, Sockets, Message Queues, Channels, Signals

- The idea is simple.

```
                    +------------------+
process 1 writes ->|                  |-> process 2 reads
                    +------------------+
```

Communication may be half-duplex or full-duplex

- Half-Duplex

```
                    +-----------------+
process 1 writes ->|                 |-> process 2 reads
                    +-----------------+
```

- Full-Duplex

```
                    +-----------------+
        writes ->|                 |-> reads
                    +-----------------+
process 1                                          process2
                    +-----------------+
         reads <-|                 |<- writes
                    +-----------------+
```

The message sent over the channel can take various forms

- the message may be just text

- the message might be a binary stream (exchange serialized objects)

- the message might be a function to call (Remote Procedure Call)

- We will focus on simple messages for now and look at data serialization later.

The exact semantics of these communciation channels varies by platform

- Examples on Unix:

  - Pipes (Anonymous and Named)

  - Sockets

  - Message Queues

- Examples on Windows:

  - Pipes (Anonymous and Named)

  - Sockets

  - COM

A central tenant of Unix systems is that programs should be small and composable.

- The primary way this is done is using Pipes.

- Example: search through all my CMake based projects looking for ones that expect a test to fail.

- `find ~ -name CMakeLists.txt -print | xargs grep WILL_FAIL`

A central tenant of Unix systems is that programs should be small and composable.

- The | character is a pipe. It connects the standard output of one program

  to the standard input of the next, forming a simple half-duplex channel.

```
                +------------------+
find stdout ->|                  |-> stdin of xargs
                +------------------+
```

Example: Unix Pipes

- `#include <unistd.h>`

- create a pipe with pipe giving two integer file descriptors

- read from one, write to the other

- See example code: `unix_pipe/`

  - pipe, pipe2 - create pipe

That's dumb, what is that good for?

- We combine that with `fork`, which makes a copy of the current process,

  called the *child*.

- The *parent* and *child* process communicate over shared pipe descriptors.

- See example code: `unix_fork/`, `unix_fork_pipe/`

  - [fork - create a child process](#)

A cross-platform solution using Qt

- The process on Unix using sockets is similar to pipes.

- Both have rough equivalents on Windows.

- Lets use Qt to do it cross-platform

QProcess

- `QProcess` is a Qt class that abstracts a process. You can start them, hijack stdin and stdout, and get their exit status.

- See example code: `qprocess_example/`

  - [QProcess Class](#)

QLocalSocket

- `QLocalSocket` is a class the abstracts a local socket. On Windows this is a named pipe and on Unix this is a local domain socket, but it does not matter to us. The idea is to use QProcess similar to fork and use QLocalSocket to setup the communication.
- See example code: `qlocalsocket_example/`
  - [QLocalSocket Class](#)

Case Study: Message Passing Interface (MPI)

- High-Performance Computing is all about leveraging multiple processing units, be they cores, CPUs, or multiple machines. One approach uses seperate processes that communicate over sockets by passing messages.

- This is standardized as MPI (there are a few different implementations).

Next Actions and Reminders

- Read about Qt shared memory