Programovanie v operačných systémoch

09 - Asynchronous programming

Jozef Šiška

Department of Applied Informatics
Comenius University in Bratislava

2016/2017

Asynchronous programming

- using asynchronous operations in a (usually) single threaded program
  - non-blocking IO and `select` / `epoll`
  - timers
  - events
- mainloop + event registration
- UI, highly performant IO servers

comparison to threads

Advantages

- simpler implementation when too many things interact (no locking, no need to think about what runs where)
- more efficient in highly asynchronous loads (server IO) - no kernel roundtrips for scheduling, synchronization

Disadvantagges

- harder / more complicated to write without a good framework / library
- everything needs to be asynchronous - one blocking operation breaks everything (dns resolving?, big local files)
  - can be solved by "pushing" the operation to another thread, but can be complicated without good libraries>
- can't utilize multiple core efficiently by default, but good frameworks can...
  - ... and synchronization can still be avoided most of the timee
- "inversion of control" - again can be helped with good framework or language design

Implementation

- cancellations, bookkeeping of "subscriptions"
- "callback hell" (older javascript, C), breaks code flow
  - lambdas, inline function syntax helps a bit, but not much
- listeners, observer pattern (OOP) - a lof of boilerplate, breaks code flow
- signals and slots
- futures, then-ables, continuations
- async frameworks - async / await in C# and javascript (and python?)
  - can be "emulated" in any language that can save and resume function execution, such as generator functions

Chat Qt / async, part 1

```
int main(int argc, char* argv[])
{
  QCoreApplication app(argc, argv);
  std::set<QTcpSocket*> clients;

  QTcpServer server(&app);
  if (!server.listen(QHostAddress::Any, 1234)) {
    throw std::runtime_error("Can't listen");
  }

  QObject::connect(&server, &QTcpServer::newConnection, [&]() {
    QTcpSocket *c = nullptr;
    while (c = server.nextPendingConnection()) {
      addClient(clients, c);
    }
  });

  qDebug() << "Listenning...";
  return app.exec();
}
```

Chat Qt / async, part 2

```
void addClient(std::set<QTcpSocket*> &clients, QTcpSocket *c)
{
  QObject::connect(c, &QAbstractSocket::disconnected, c, [c, &clients]() {
    clients.erase(c);
    c->deleteLater();
  });

  QObject::connect(c, &QIODevice::readyRead, [c, &clients]() {
    auto data = c->readAll();
    for( auto wc : clients) {
      if (wc->bytesToWrite() < 10*1024*1024) {
        wc->write(data);
      }
    }
  });
  clients.insert(c);
}
```