Error Handling in Multiuser Systems

A Software Engineer's Perspective

Sean Kerwin

Who Am I?

- Sean Kerwin
- FGCU ‘07
- Architect at INgage Networks
    - First social networking site to become a TV show!
    - American Express OPEN forum site

Why Listen to Me?

- Curse of our field: the really interesting stuff can’t be taught algorithmically
- Failure is the best teacher
  - Other people’s failures tend to be more entertaining

Why NOT Listen to Me?

- Trust but verify
- I firmly believe: being a good developer requires healthy skepticism
- Even if all I accomplish is to tick you off enough that you set out to prove me wrong about something, that’s pretty cool

Why Error Handling Matters

- Therac-25
- Serious software bug
  - Six people got radiation overdoses
  - Three deaths

What Am I Going to Talk About?

- What do I mean by multiuser system?
- How do they differ from the desktop?
- What's an error?
- Avoiding errors
- General error-handling wisdom

What's a Multiuser System?

- Websites are an easy answer
- APIs powering mobile applications
- The servers running AIM, ICQ, etc.
- Networked RDBMS
- The World of Warcraft servers
- Amazon's cloud services

How Are They Different?

- Control the capacity of one client to affect the experience of others
  - Control, not prevent. Usually.
  - The classic error handling techniques no longer works

What’s an Error?

- ‘Exception’ and ‘error’ are not synonyms!
- Defining what is and isn't an error state is an important part of your design
  - But it will probably evolve
- Know your library or runtime
- Define expectations for your system

Know Your Library or Runtime

- Is an exception an error?
- Is an error code an error?

- Socket programming in .NET is instructive
  - Exceptions AND codes, and both can be either errors or perfectly expected

Define Expectations

- Classes and method signatures are contracts; be specific.
  - Can this parameter be null?
- Distinguish ‘programmer errors’ from ‘user errors’.
- Keep an eye out for ‘fundamental assumption errors’.

Expectations == Specifications?

- Yes. And no. And yes!
- And no.

Best Error Handling Approach?

Avoid them entirely.

Avoiding Errors

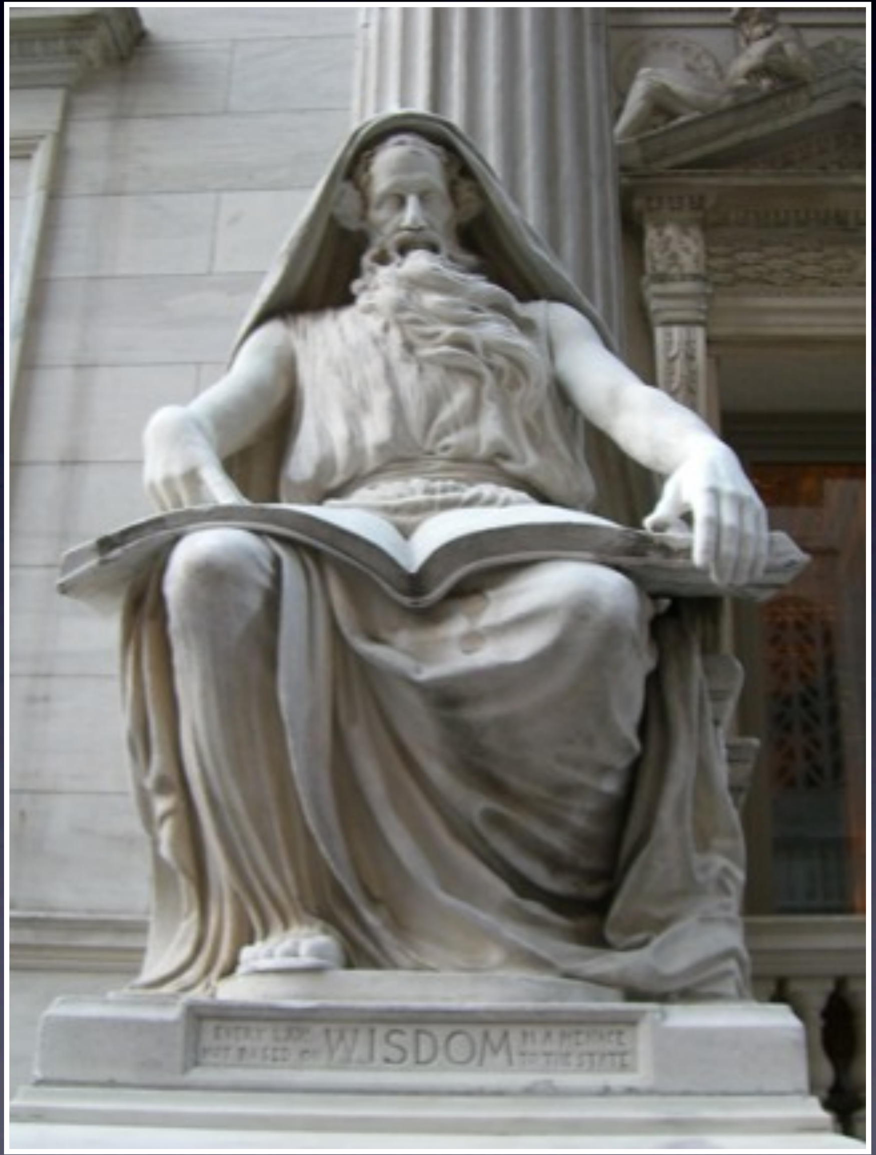
- Remove *opportunities* for error
- Properly structured code / unit testing
- Have a big toolkit:
  - Use strong typing to your advantage
  - Use functional styles to your advantage
  - Use immutability to your advantage

Error-Averting Patterns

- Know the options for your language
- Template method pattern
  - Know how to build a base class *right*
- Type-safe enum idiom

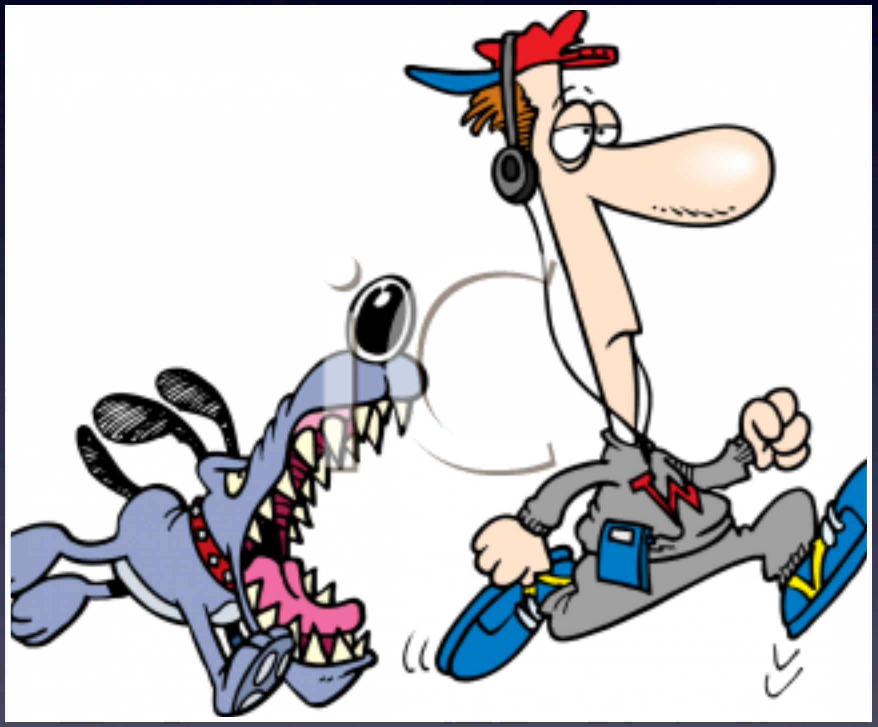
General Error-Handling Wisdom

- Know when you're failing
- Fail safely
- Fail fast
- Avoid single points of failure
- Log Intelligently



Wisdom: Know You're Failing

- Locally:
  - Check return values
  - Know what's allowed
- Globally:
  - Varies by platform/framework/language/library/etc.



Wisdom: Failing Safely

- An electronic lock has a serious failure. What does it do?
- More relevant: authentication in your application

```
function authentication_is_valid( uid, password ) {
   var identityRecord = database_load(uid);


   if ( !valid_password(identityRecord, password ))
      return false;

   if ( identityRecord.isBlocked )
      return false;

   if ( ! identityRecord.allowsRemoteLogin )
      return false;


   return true;
}
```

```
function authentication_is_valid( uid, password ) {
  var identityRecord = database_load(uid);

  if ( identityRecord ) {
    if ( !valid_password(identityRecord, password ))
      return false;

    if ( identityRecord.isBlocked )
      return false;

    if ( ! identityRecord.allowsRemoteLogin )
      return false;
  }

  return true;
}
```

Wisdom: Fail Fast

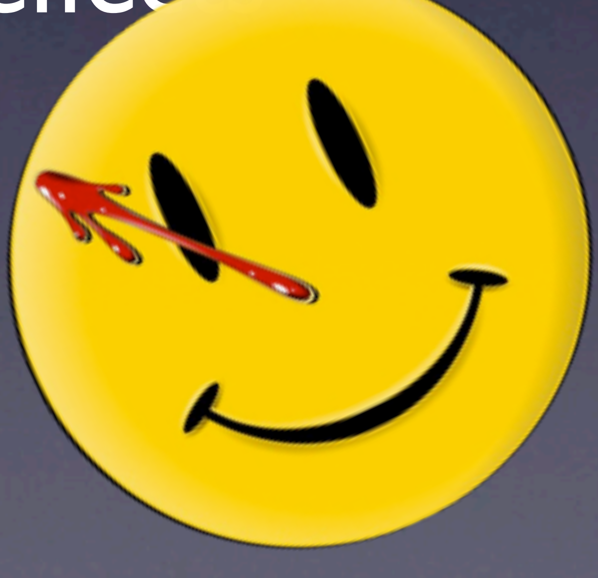
- If it’s written right, `valid_password` is slow.
  - Do it last!
- Avoids un-needed work, but also allows for more specific/useful errors
- Also aids in keeping a consistent state

Wisdom: Single Points of Failure

- Redundant web servers and DBs, but single router
- Large cluster with one ‘manager’ node
- Sometimes unavoidable?

Wisdom: Log Intelligently

- Eventually log items become action items
  - Respond intelligently
- Don't just log errors
  - But understand performance effects
- Who watches the watchmen?



Conclusion

- If you have multiple users, the bar is higher
- Don’t think of error-handling as ancillary
- Use the tools available to reduce risk
- Know you’re failing, and do it safely, quickly, rarely, and loudly.

Questions?

How many surrealists does it take to screw in a light bulb?