So, you’ve decided to write software for a living.
So, you’ve decided to write software for a living.

Good for you!
This is definitely the right path for you if:
This is definitely the right path for you if:

you are clever and industrious
This is definitely the right path for you if:

you are **clever** and **industrious**

you like the idea of **making neat things**
that other people use
This is definitely the right path for you if:

you are **clever** and **industrious**

you like the idea of **making neat things**
that other people use

you are a pretty good **programmer**
This is definitely the right path for you if:

you are **clever** and **industrious**

you like the idea of **making neat things**

that other people use

you are a pretty good **programmer**

you **hate people** and want to **hide** behind a computer
This is definitely the right path for you if:

- you are **clever and industrious**
- you like the idea of **making neat things** that other people use
- you are a pretty good **programmer**
- you **hate people** and want to **hide** behind a computer
- you want a **predictable job** with low stress and plenty of **time off**
(the previous slide contains two errors)
This is definitely the right path for you if:

you are clever and industrious

you like the idea of making neat things that other people use

you are a pretty good programmer

you hate people and want to hide behind a computer

you want a predictable job with low stress and plenty of time off
This is definitely the right path for you if:

- You are clever and industrious.
- You like the idea of making neat things that other people use.
- You are a pretty good programmer.
- You hate people and want to hide behind a computer.
- You want a predictable job with low stress and plenty of time off.
But...I really hate people!
But...I really hate people!

(That’s why I got into CS in the first place!)
But...I really hate people!

(That's why I got into CS in the first place!)
Building software is all about people.

You will spend as much time dealing with your colleagues as with your compiler.

Planning new projects

Designing systems to solve problems

Arguing for your proposed solution (and the necessary time/people to get it done)

Explaining your system to others who will need to understand it deeply

Listening to your manager, peers, marketing department, testing team, users, …

Pushing back on all those parties when necessary
The Players
The Players

YOU
The Players

YOU

Your team
The Players

- Your manager
- You
- Your team
The Players

His director

Your manager

YOU

Your team
The Players

His director
Your manager
YOU
Your team
Other teams
The Players

- His director
- Your manager
- QA/Test
- You
- Your team
- Other teams
The Players

- His director
- Your manager
- QA/Test
- YOU
- Your team
- Sales/Marketing
- Other teams
The Players

- His director
- Your manager
- Sales/Marketing
- QA/Test
- You
- Your team
- Users
- Other teams
The Players

- His director
- Your manager
- Sales/Marketing
- QA/Test
- YOU
- Your team
- Other teams
- Clients
- Users
The Players

- The CEO
- QA/Test
- The CEO
- QA/Test
- Users
- Clients
- Partners
- Your team
- Other teams
- Sales/Marketing
- His director
- Your manager
- YOU
Lesson:
Learn to communicate.
Lesson: Learn to communicate.

Corollary: Play well with others.
What else should you know?

The following comes from:
My experience
My friends’ experience
The Internets

Your mileage will vary
but a lot of this stuff is the same everywhere
The Release Cycle
“We’ll ship it when it’s done.”
“We’ll ship it when it’s done.”
“We’ll ship it when it’s done.”
Development
Development

Release
Development → Release → Party!
Redesign

Design

Development

Test

Release

Party!
Redesign

Bugs

Design

Bug Fixes Only

Development

Test

Release

Party!
Really bad bugs

Redesign

Bugs

CODE FREEZE (stabilization)

Really bad bugs

Redesign

Bugs

CODE FREEZE (stabilization)
Really bad bugs

Feature Wishlist

Design

Bug Fixes Only

Test

Release

Party?

Redesign

Bugs

CODE FREEZE (stabilization)
Tools
Source Control
Source Control

* We’ve talked about this before
  (Remember the bacon?)

* As projects grow, coordinating source becomes more complex

* If you interview with a company that doesn’t use source control:

  KEEP WALKING
Tags & Branches

time flows to the right
Tags & Branches

“the trunk”

time flows to the right
Tags & Branches

“the trunk”

time flows to the right
Tags & Branches

tag: “version 0.1”

“the trunk”

time flows to the right
Tags & Branches

time flows to the right

“the trunk”

“branch”

tag: “version 0.1”

time flows to the right
Tags & Branches

time flows to the right

“the trunk”

“branch”

tag: “version 0.1”

time flows to the right
Tags & Branches

tag: "version 0.1"

time flows to the right
Tags & Branches

tag: “version 0.1”

time flows to the right
Tags & Branches

time flows to the right

"the trunk"

"branch"

tag: "version 0.1"
Tags & Branches

"the trunk"

tag: "version 0.1"

"branch"

integration

time flows to the right
Tags & Branches

"the trunk"

tag: "version 0.1"

time flows to the right

"branch"

integration
BeOS branch structure

main
BeOS branch structure

rel

main
BeOS branch structure

- rel
- main
- exp
BeOS branch structure
BeOS branch structure
BeOS branch structure

rel
main
exp

2.0b1 2.0b2
BeOS branch structure

rel

main

exp
PalmOS branch structure
PalmOS branch structure

Not shown:
~10 sub-branches of
appservices, coretech
Code
Reading Code

* Analogues: Cryptography, archaeology, Talmud interpretation†
* The original author is not always available
* Best performed in pairs

† [http://www.joelonsoftware.com/articles/fog0000000053.html](http://www.joelonsoftware.com/articles/fog0000000053.html)
Some code to read
Example #1:
StrEndsWith(char * str, char * suffix)
Example #1:
StrEndsWith(char * str, char * suffix)

Boolean StrEndsWith(CharPtr string,
                        CharPtr endString)
{
    CharPtr p1;
    CharPtr p2;

    p1 = string;
    p2 = endString;

    if (!p1 || !p2)
        return false;

    p1 = p1 + StrLen(p1) - StrLen(p2);

    return false;
}
Example #2: How done is done?
Example #2: How done is done?

```c
Boolean IMAP4getMailBodyInt(ULong uid, mailMessage*msg, callbackFn callback,
    CharPtr file, CharPtr boundary, int encoded,
    Boolean binaryAttach, Boolean isRead, Boolean multiPartAlternative,
    Long aType)
{
    VoidHand bufHand = 0;
    CharPtr bufPtr = 0;
    CharPtr p1 = 0;
    CharPtr p2 = 0;
    CharPtr receivePtr = 0;
    int bytesRead = 0;
    int bytesSoFar = 0;
    Boolean completed = false;
    Boolean cancel = false;
    Boolean retVal = false;

    Boolean truncated = false;
    int msgNum = 0;
    CharPtr file2 = 0;
    CharPtr boundary2 = 0;
    Boolean omittedAttachment = false;
    Boolean done = false;
    Boolean readable = isRead;
    int isEncoded = encoded;
    VoidHand curHand = 0;
    Boolean Done = false;
    Long progressBytesSoFar = 0;

    #ifdef OLDWAY
    Boolean oldIMAP = false;
    #endif
    Boolean stop = false;
    [...]
Example #2: How done is done?

```c
Boolean IMAP4getMailBodyInt(ULong uid, mailMessage* msg, callbackFn callback, CharPtr file, CharPtr boundary, int encoded, Boolean binaryAttach, Boolean isRead, Boolean multiPartAlternative, Long aType)
{
    VoidHand bufHand = 0;
    CharPtr bufPtr = 0;
    CharPtr p1 = 0;
    CharPtr p2 = 0;
    CharPtr receivePtr = 0;
    int bytesRead = 0;
    int bytesSoFar = 0;
    Boolean completed = false;
    Boolean cancel = false;
    Boolean retVal = false;

    Boolean truncated = false;
    int msgNum = 0;
    CharPtr file2 = 0;
    CharPtr boundary2 = 0;
    Boolean omittedAttachment = false;
    Boolean done = false;
    Boolean readable = isRead;
    int isEncoded = encoded;
    VoidHand curHand = 0;
    Boolean Done = false;
    Long progressBytesSoFar = 0;

    #ifdef OLDWAY
    Boolean oldIMAP = false;
    #endif

    Boolean stop = false;
    [...]
Example #2: How done is done?
Example #2: How done is done?

Boolean completed = false;
Boolean cancel = false;
Boolean retVal = false;
Boolean done=false;
Boolean Done=false;
Boolean stop=false;
Example #2: How done is done?

```java
Boolean completed = false;
Boolean cancel = false;
Boolean retVal = false;
Boolean done = false;
Boolean Done = false;
Boolean stop = false;

Done = ... 
...
while (!Done) {
    ...
    ...;
}
```
Example #2: How done is done?

```java
Boolean completed = false;
Boolean cancel = false;
Boolean retVal = false;
Boolean done = false;
Boolean Done = false;
Boolean stop = false;

Done = ...
...
if (!done)
{
    ...
}
while (!Done) {
    ...
}
```
Example #2: How done is done?

```java
Boolean completed = false;
Boolean cancel = false;
Boolean retVal = false;
Boolean done = false;
Boolean Done = false;
Boolean stop = false;

Done = ...
...
if (!done)
{
  ...
}
while (!Done) {
  ...
    
    if (stop)
        break;
}
Example #2: How done is done?

```java
Boolean completed = false;
Boolean cancel = false;
Boolean retVal = false;
Boolean done = false;
Boolean Done = false;
Boolean stop = false;

Done = ... 
...
if (!done)
{
  ...
  ...
}
while (!Done) {
  ...
  if (stop)
    break;
  ...
  stop = true;
  cancel = true;
  done = true;
}
```
How do you keep code like this out of your program in the first place?
Code Reviews

- Also known as: “the Spanish Inquisition”
- Learn to love them—the best software shops swear by them
- An opportunity for you to stop others’ broken code before it hits the system
- An opportunity for others to stop your broken code before it hits the system
Hidden features of code reviews

- Someone else has seen your code!
- This means you have help when something breaks
- (Also: your teammates aren’t screwed if you flake)
Automatic Builds

* Build server: tireless, dauntless, indefatigable
* Every hour, or every $N$ checkins, or whatever:
  * Check everything out
  * Compile it all
  * Run a subset of unit tests, viz. “sniff tests”
    * ELEC majors can explain this terminology
* If it doesn’t compile, or doesn’t pass the tests…
YOU BROKE IT!
YOU BROKE IT!

(now fix it, genius)
Here’s a bad idea
Here’s a bad idea

* 4:45 PM:
Here’s a bad idea

* 4:45 PM:
  * check in a bunch of shiny new code
Here’s a bad idea

• 4:45 PM:
  • check in a bunch of shiny new code

• 5:00 PM:
Here's a bad idea

* 4:45 PM:
  * check in a bunch of shiny new code

* 5:00 PM:
  * go home
Here’s a bad idea

✶ 4:45 PM:
   ✶ check in a bunch of shiny new code

✶ 5:00 PM:
   ✶ go home
Here’s a bad idea

• 4:45 PM:
  • check in a bunch of shiny new code

• 5:00 PM:
  • go home

• (applies in this course as well)
You WILL get the build hat.
You WILL get the build hate
Testing
Who are QA*?

*“Quality Assurance”

* The front line in the War On Bugs
* Professional program-crashers
* You may be one of these people!
* Some people are incredibly gifted at breaking things (this usually takes a fair bit of coding experience)
* They will bring your code to its (metaphorical) knees
* What are they looking for?
Bugs
Sources of bugs

- Misunderstanding
- Lousy code
- Unforeseen consequences
- Sleep deprivation
Dan’s bug stories

- The Case of the Christmas Lights
- The Case of the Time-Bomb MP3 Stream
Bug Tracking
Bug Tracking

- An essential part of keeping track of all the broken, half-implemented, flaky bits of your code
Bug Tracking

- An essential part of keeping track of all the broken, half-implemented, flaky bits of your code
- A glorified “to-do” list
An essential part of keeping track of all the broken, half-implemented, flaky bits of your code

A glorified “to-do” list

Bugs have assignees, attachments, running comments
Bug Tracking

* An essential part of keeping track of all the broken, half-implemented, flaky bits of your code
* A glorified “to-do” list
* Bugs have assignees, attachments, running comments
* Bugs have states: “open”, “fixed”, “unreproducible”, etc.
Bug Tracking

* An essential part of keeping track of all the broken, half-implemented, flaky bits of your code
* A glorified “to-do” list
* Bugs have assignees, attachments, running comments
* Bugs have states: “open”, “fixed”, “unreproducible”, etc.
* Each one is like an email or BB thread
Bug Tracking

✽ An essential part of keeping track of all the broken, half-implemented, flaky bits of your code

✽ A glorified “to-do” list

✽ Bugs have assignees, attachments, running comments

✽ Bugs have states: “open”, “fixed”, “unreproducible”, etc.

✽ Each one is like an email or BB thread

✽ It will become your homepage
Bugzilla Bug 3
Red widgets don't appear.

Bug List: (1 of 1) First Last Prev Next Show list Query page Enter new bug

Bug: 3
Product: Widget Browser
Component: User Interface
Status: NEW
Resolution: 

Platform: PC
OS: Windows 98
Version: 1.1
Priority: P2
Severity: normal

Assigned To: maty@chariot.net.au (Matthew Tuck)
Target Milestone: ---

CC:

QA Contact:
URL:

Summary: Red widgets don't appear.

Status Whiteboard:
Keywords:

Attachment
Type
Modified
Status
Actions
Screenshot of the red widgets not appearing
application/octet-stream
2002-07-12 16:52
none
Edit
Create a New Attachment
(proposed patch, testcase, etc.)

Bug 3 depends on:
Show dependency tree

Bug 3 blocks:
Show dependency graph

Additional Comments:

* Leave as NEW
* Accept bug (change status to ASSIGNED)
Case 10: Error 293 printing to laser printer

ApartmentSwapper - Misc

Status: Closed: Resolved (Fixed) - Add Release Note
Assigned To: CLOSED
Category: Bug
Fix For: Unreleased
Priority: 1 - Must Fix
Due: 3/15/2006 5:00 PM
Version:
Computer:

Opened by Henry Wadsworth 3/15/2006 2:50 PM

Repro steps:
Run ApartmentSwapper
Navigate to apartment 217
Click "Show Downstairs"
File I Print
(Default printer is Dell 1700 Laser Printer.)
BUG: following error occurs; nothing is printed

Every case is always assigned to exactly one person until it is closed. That person is responsible for it until they assign it to someone else. Here's a typical bug. FogBugz keeps a detailed, chronological history of each case.

Error 293

An error occurred. I do not know why or where.
Lazy programmer.


Assigned to Myrna Loy by Maureen O'Sullivan 3/15/2006 2:53 PM

Myrna, the Haku error message sounds like you :) Please get this fixed today because I know that Celebration Apartments just got in a new shipment of these laser printers.

Resolved (Fixed) by Myrna Loy 3/15/2006 3:05 PM

Looks like error 293 should have read "Printer out of paper." I've changed the error message wording in the code, and I've put some paper in your printer!

Closed by Henry Wadsworth 3/15/2006 3:38 PM

That fixed it.

Subscribe to receive email when this case changes.
### List of issues

<table>
<thead>
<tr>
<th>ID</th>
<th>Activity</th>
<th>Title</th>
<th>Status</th>
<th>Creator</th>
<th>Assigned To</th>
</tr>
</thead>
<tbody>
<tr>
<td>468</td>
<td>2 minutes ago</td>
<td>Donec consequat convallis quam.</td>
<td>unread</td>
<td>admin</td>
<td>epsilon</td>
</tr>
<tr>
<td>288</td>
<td>2 minutes ago</td>
<td>Vivamus tincidunt.</td>
<td>done-cbb</td>
<td>admin</td>
<td>demo</td>
</tr>
<tr>
<td>228</td>
<td>2 minutes ago</td>
<td>Donec consequat convallis quam.</td>
<td>done-cbb</td>
<td>admin</td>
<td>beta</td>
</tr>
<tr>
<td>136</td>
<td>2 minutes ago</td>
<td>Suspendisse et turpis.</td>
<td>testing</td>
<td>admin</td>
<td>epsilon</td>
</tr>
<tr>
<td>99</td>
<td>2 minutes ago</td>
<td>Donec consequat convallis quam.</td>
<td>deferred</td>
<td>admin</td>
<td>epsilon</td>
</tr>
</tbody>
</table>

**urgent**

<table>
<thead>
<tr>
<th>ID</th>
<th>Activity</th>
<th>Title</th>
<th>Status</th>
<th>Creator</th>
<th>Assigned To</th>
</tr>
</thead>
<tbody>
<tr>
<td>477</td>
<td>2 minutes ago</td>
<td>Vestibulum gravida.</td>
<td>deferred</td>
<td>admin</td>
<td>admin</td>
</tr>
<tr>
<td>472</td>
<td>2 minutes ago</td>
<td>Sed convallis vehicula felis.</td>
<td>deferred</td>
<td>admin</td>
<td>beta</td>
</tr>
<tr>
<td>355</td>
<td>2 minutes ago</td>
<td>Fusce pede enim, nonummy sit amet, dapibus a, blandit eget, metus.</td>
<td>done-cbb</td>
<td>admin</td>
<td>admin</td>
</tr>
<tr>
<td>289</td>
<td>2 minutes ago</td>
<td>Aenean i</td>
<td>urus sed, ege</td>
<td>sed, ege</td>
<td></td>
</tr>
<tr>
<td>282</td>
<td>2 minutes ago</td>
<td>Nam eget</td>
<td>urus sed, ege</td>
<td>sed, ege</td>
<td></td>
</tr>
<tr>
<td>196</td>
<td>2 minutes ago</td>
<td>Integer te</td>
<td>urus sed, ege</td>
<td>sed, ege</td>
<td></td>
</tr>
<tr>
<td>181</td>
<td>2 minutes ago</td>
<td>Nam odio</td>
<td>urus sed, ege</td>
<td>sed, ege</td>
<td></td>
</tr>
<tr>
<td>175</td>
<td>2 minutes ago</td>
<td>Integer te</td>
<td>urus sed, ege</td>
<td>sed, ege</td>
<td></td>
</tr>
</tbody>
</table>

**Issue468 Editing**

**Issue 468 assigned to edited ok**

- **Title**: Donec consequat convallis quam.
- **Priority**: critical
- **Status**: deferred

**File**

**Change Note**

*Created on 2004-05-06, 01:51:28 by admin, last changed 2004-05-06, 02:03:06 by demo*

**History**

<table>
<thead>
<tr>
<th>Date</th>
<th>User</th>
<th>Action</th>
<th>Args</th>
</tr>
</thead>
<tbody>
<tr>
<td>2004-05-06 02:03:06</td>
<td>demo</td>
<td>set</td>
<td>assignedto alpha -&gt; delta</td>
</tr>
<tr>
<td>2004-05-06 02:03:06</td>
<td>demo</td>
<td>set</td>
<td>assignedto beta -&gt; alpha</td>
</tr>
<tr>
<td>2004-05-06 01:51:46</td>
<td>demo</td>
<td>set</td>
<td>assignedto epsilon -&gt; beta</td>
</tr>
<tr>
<td>2004-05-06 01:51:46</td>
<td>demo</td>
<td>set</td>
<td>status unread -&gt; deferred</td>
</tr>
<tr>
<td>2004-05-06 01:51:28</td>
<td>admin</td>
<td>create</td>
<td></td>
</tr>
</tbody>
</table>
Schedules

✽ Estimation is hard
✽ Nobody has solved this

✽ Tips:
  ✽ Make a schedule
  ✽ Break things up into small tasks (between 1 hr and 1 day)
  ✽ Track how well you did!
  ✽ This is the only way to improve your estimating skills
Wrap-up

• A look inside the sausage software factory
• No “best practices” in software, yet
  • Just “common practices”
• These examples come from my experience; yours will be different in many ways
• “How to make software” reading list
  • Posted on the course website as the semester goes on