The wheel already exists...

EVENT AND SIGNAL DRIVEN PROGRAMMING TECHNIQUES

CS Lesson
Lecture
Rant
Tutorial
Talk

FEEDBACK IS PRECIOUS

Patterns and Concepts – oh my

IN THE BEGINNING – CS 201

STRUCTURED PROGRAMMING

- simple, hierarchical program flow structures
  - sequence
  - selection
  - repetition
- procedural
  - adds sub-routines or functions
- object oriented
  - modularizes components of code

EVENT DRIVEN PROGRAMMING

TRANSITIONING TO EVENTS

- code flow is controlled by the user, not the program
- software can sit “idle” until an event occurs
- allows software to react
- why?

PUBLISH/SUBSCRIBE

- type of event programming
- this is NOT observer
- publisher and subscriber are decoupled
- subscribers express interest in a state change/event/message/signal
- each subscriber receives a copy

SUBJECT/OBSERVER

- this is a subset of publish/subscribe NOT the other way around
- in this case, the subject maintains the observers and notifies them of state change/event/message/signal
- more tightly coupled then publish/subscribe

EVENT/HANDLER

- events are spawned by the system
- events are dispatched to handlers which are attached to them
- handler can go by many names – callback, closure, function
- events might have priority, might interrupt depending on the dispatch method used

SIGNAL/SLOT

- signal – something you know is going to happen that is attached to a class
- slot – method in a class that can be connected to a signal
- connect a signal to a slot (wire it up)
- emit the signal
- requires oo

ASYNCRONOUS VS. SYNCHRONOUS

- event-driven !== asynchronous
- structured !== synchronous

- concurrency through
  - *Async* I/O (epoll, IOCP, kqueue, et al)
  - threads
  - forking (multi-process)

(MAIN) EVENT LOOP

- poll event provider
- blocks until event arrives
- calls the event handler (dispatches)
- main is added if it is the highest level of control in a program

DISPATCHER

- gets event
- determines which handlers get called
  - either attached
  - determined by type
  - stored in some manner
- calls the handlers

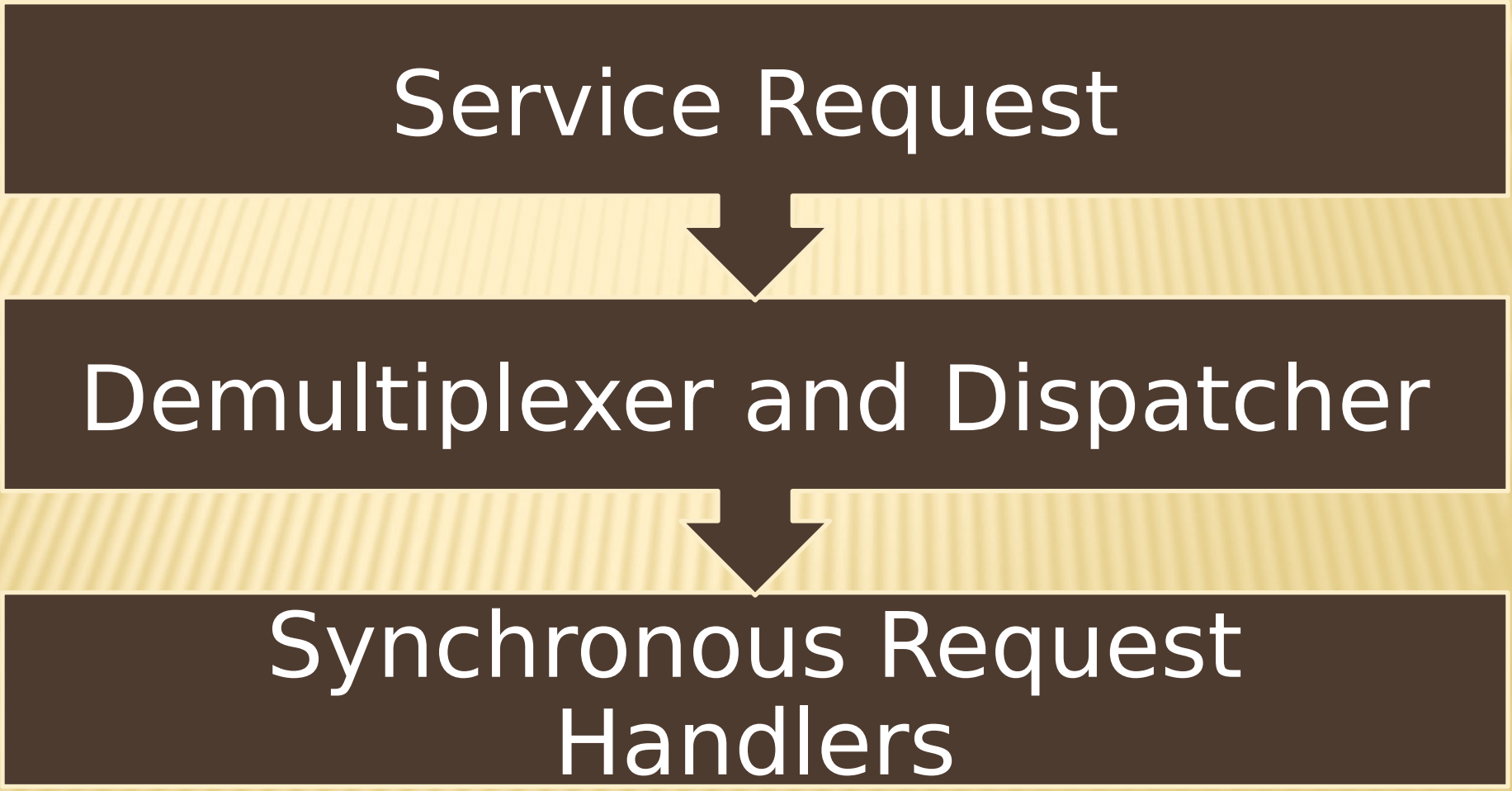
MESSAGE QUEUE

- Buffers the input stream of events
- FIFO usually, unless priority or other features are present in the queue
- while(events_pending)

EVENT DRIVEN STATE MACHINE

- A number of behavioral nodes waiting for a trigger to execute the transition
- The trigger in this case is an event

REACTOR

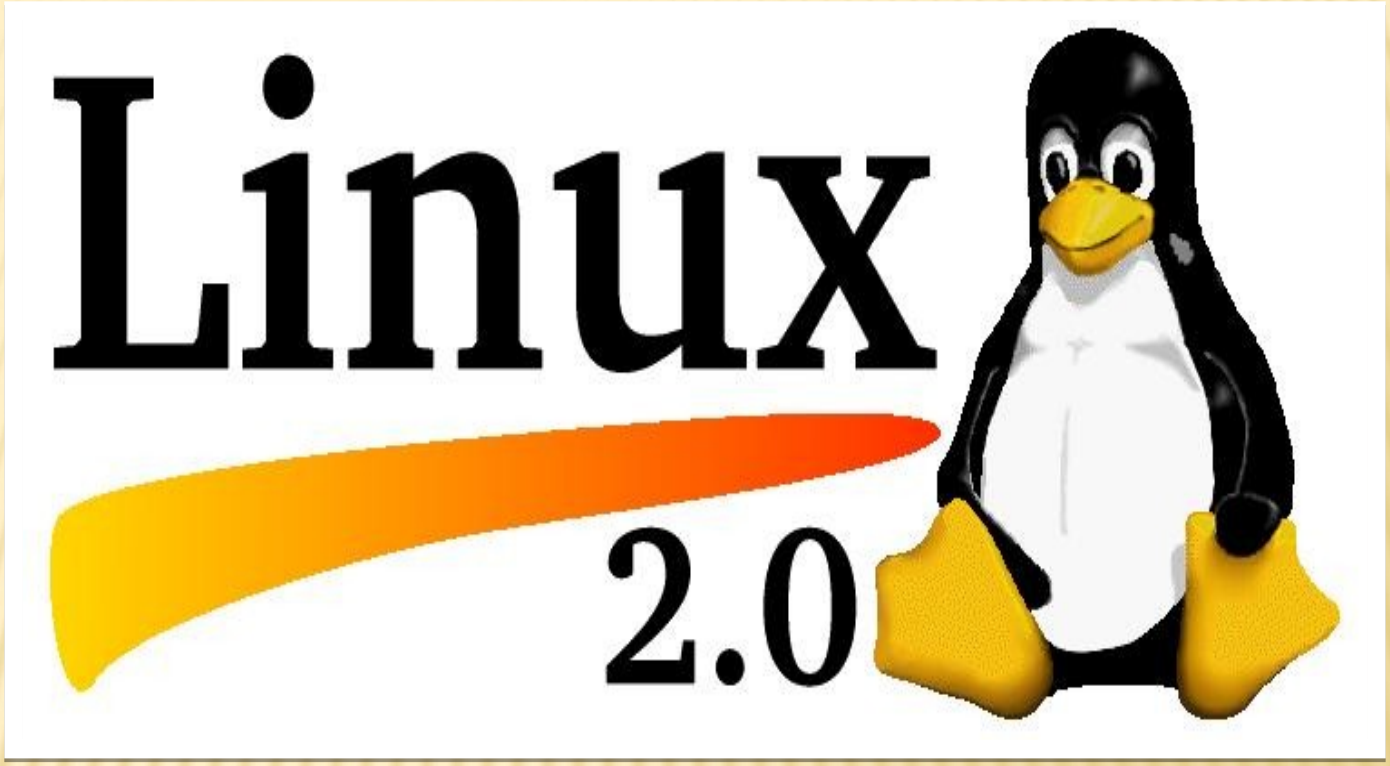

Service Request
Demultiplexer and Dispatcher
Synchronous Request Handlers

INTERRUPTS

- asynchronous signal/event indicating need for change in execution or need for attention
- “The ship is going to explode”
- A *signal* is a software interrupt delivered to a process

The Wheels keep on Turning

WE’VE BEEN DOING THIS ALL ALONG



KERNEL SIGNALS

“Get off my Lawn!”

USING IT WITH PHP

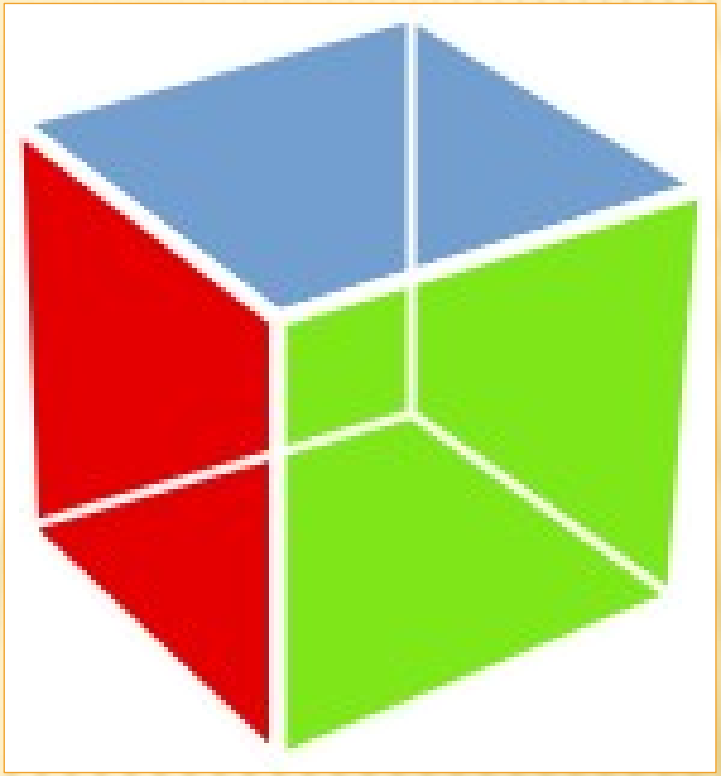
- PCNTL
- very simplistic
- most signals (other then SIGUSR1 and SIGUSR2) have predefined meanings

XLIB SIGNALS

1985 called and they want their shoes.

XLIB AND XNEXTEVENT

- blocks until event appears in the queue
- application process it
- only handles window system events
- calling xlib functions from a signal handler “POOF”



GOBJECT (FORMERLY GTK+)

Gimp wanted a nice toolkit and had a gnome

GOBJECT IN PHP

- PHP-GTK
- PHP-GTK (next)

- Extend gobject class
- define your signals
- attach to your signals
- emit your signals

FEATURES

- Register Signal
- Connect Signal to Closure
- Emit Signal
- User Data
- Accumulators
- Details

GOBJECT EVENT LOOP

- Manages all sources of events
- Event type can be added
- Gmaincontext for each thread to manage sources
- Priorities for event sources and idle functions

QT

For those who like a little ++ with their C

QT – “EVENTS” AND “SIGNAL/SLOT”

- event – virtual function in a class you reimplement to handle the event
- signal – wired to a callback (slot) by a metaobject class, no loop necessarily required

WIN API

Pump that message baby

.NET

The windows api, memory managed and less sucky

LIBEVENT (NODE.JS, TWISTED, EVENTMACHINE…)

It’s all in the family!

Deep Thoughts on Good Code

BEST PRACTICES

THE LEGACY OF HTTP

WHY EVENTS OR SIGNALS?

IMPORTANCE OF CLOSURES AND TRAITS

NAME IT PROPERLY

- Publish / Subscribe
  - Decoupled
- Subject / Observer
  - stateless
- Event / Handler
  - event loop
- Signal / Slot
  - Tightly coupled objects
  - Traits
- Dispatcher
- (Main) Event Loop
- State Machine
- Message Pump
- Reactor

STANDARD INTERFACES

- Observer? Use SplSubject and SplObserver
- Pub/Sub?
- Event/Handler?
- Signal/Slot?

BORROW FROM OTHER LANGUAGES

- Any C based language translates REALLY well to PHP
- If you can’t depend on an extension, but want to “drop it in” – do the same API in a different namespace and class_alias as appropriate
- Upgrading from stone wheels to vulcanized rubber is great, but don’t reinvent

Event Driven Code in Action

IN THE WILD

ZF2

- EventManager (aggregator, trigger events)
- Listener (callback)
- Event (action modeled as object)

SYMFONY COMPONENTS – EVENT DISPATCHER

- Register of listeners
- Event object
- notify, notifyUntil, filter

ZETA COMPONENTS

- Component – SignalSlot
- signal as event, slot as callable
- priorities
- static connection (hook)

ALMOST BUT NOT QUITE

- Lithium
- Prado
- Wordpress
- Symfony 1

THE FUTURE?

- Best new hope for clean, easy eventing implementations? Traits
- Some good wrappers around existing C libraries (libevent, dbus, glib/gobject, qt, winapi, .NET)
- Pulling in good ideas from other languages, PHP evolves

RESOURCES

- http://components.symfony-project.org/event-dispatcher/
- http://incubator.apache.org/zetacomponents/documentation/trunk/SignalSlot/tutorial.html
- http://weierophinney.net/matthew/archives/266-Using-the-ZF2-EventManager.html
- http://c2.com/cgi/wiki?EventDrivenProgramming
- http://eventdrivenpgm.sourceforge.net/
- http://www.gtk.org/
- http://qt.nokia.com

BRING IT ON

- Good Libraries
- Blog posts
- Articles
- Extensions

CONTACT ME

- http://emsmith.net
- http://joind.in/3756
- auroraeosrose@gmail.com
- IRC – freenode – auroraeosrose
- #php-gtk #coapp and others