The

OSP 2

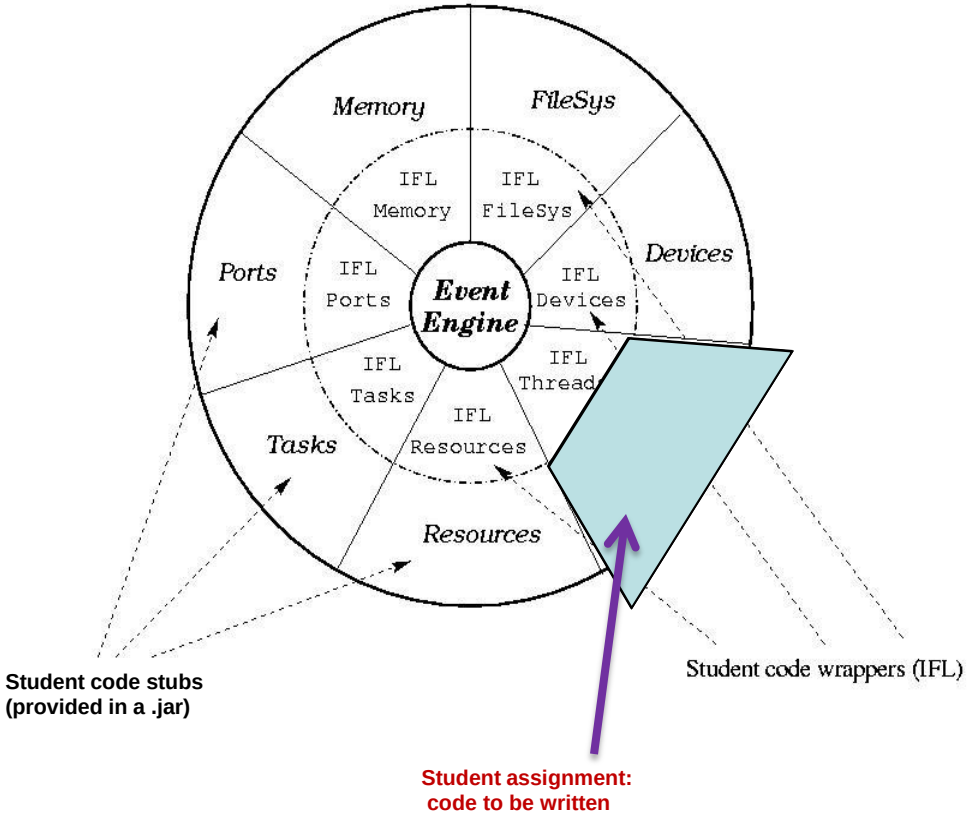
Survival Guide

What is OSP 2

- Educational platform that simulates
  - Hardware (devices, CPU, memory)
  - Events (interrupts, timer events)
  - Job streams (task/thread life cycle, I/O requests, interprocess communication, resource requests)
- Provides
  - OS modules that react to events and requests coming from the simulated job streams
- Checks
  - For semantic errors in how these OS modules handle the simulated requests (these errors guide students towards correct solutions)

The Essence of Student Projects

- A student project consists of OSP 2 with one of the OS modules taken away
- Student is given one or more template files for the classes to be implemented
- Must use the API described in the manual
- Completed student module is plugged in and run
- If OSP issues no errors or warnings – the project is implemented correctly



What Does the API Looks Like?

- Timer class (simulates timer device)
  public final static void *set*(int time)
  public final static long *get*(int time)
- Interrupt Vector class (interrupt register)
  public final static void *setInterruptType*(int interruptType)
  public final static int *getInterruptType*()
  public final static ThreadCB *getThread*()
- A lot more …

Events

- Anything that a thread might wait for is represented by an Event object
  - E.g., I/O waits are represented by IORB objects (I/O request blocks). IORB is a subclass of Event. So is PageTableEntry.
- Events have queues
  - When a thread needs to wait for I/O, its ThreadCB is enqueued to the corresponding IORB (which is an event, as we just saw)
  - When the event occurs (eg, the I/O is done) the event is "*signaled*" and the thread waiting for it is notified (awaken)
- Events are accessed via their own API
- Make sure you understand events well!

Demo.jar

- Set the CLASSPATH environment variable appropriate for your Java installation.
- You can play with the demo program

  Unix/Mac:

  java -classpath .:Demo.jar:${CLASSPATH} osp.OSP

  Or simply: make demo (requires GNU make)

  Windows:

  java -classpath .;Demo.jar;%CLASSPATH% osp.OSP

- Don’t play with the demo too much: get down to actual work

OSP.jar

- After creating the missing OSP2 module (ie, your project code), compile and run:

```
javac -g -classpath .:OSP.jar -d . *.java
java -classpath .:OSP.jar  osp.OSP
```

(on Windows: replace ":" with ";")

On Unix/Mac, can simply:

`make` (to compile)

`make run` (to compile+run)

(again, install GNU make, if Mac)

- Using Eclipse?
  - add `OSP.jar` to the project and figure out the rest

OSP.log

- Everything that happened during the execution AND that you need to know about is recorded in OSP.log
  - Will contain errors and warnings
  - This is your primary tool for debugging
    - There are other minor tools (see the manual), but OSP.log is by far the most important one
  - Will have to figure out what the errors/warnings mean and work backwards in the trace
    - This is hard and requires understanding of the OS terms and of what you are supposed to do

Obfuscation

- OSP.jar and Demo.jar code is obfuscated and cannot be meaningfully decompiled
- This means: use **only** the API described in your project chapter
  - other methods that you might find in the manual (in *other* projects' sections) will not be available for your use:
    - you will be getting compilation errors, if you try.

Quirks

- OSP2 might appear to hang after it tells you that it is finished – just kill it
- It might give an exception after it said that is has terminated – ignore
- *Disclaimer*: OSP2 is very stable, but has some minor issues. It is possible that OSP2 will report an error when there is none:
  - In some rare cases, you might be getting intermittent errors while running your project (e.g., once every 10 runs).
  - If the errors are rare and the error message seems unrelated or wrong, it probably is not your fault

How to Survive OSP 2

- Read the textbook and understand the functionality required of each module
- Read the OSP 2 manual to know the API
  - OSP 2 manual is NOT a replacement for the textbook
    - Nor is the textbook a replacement for the manual!!
  - The OSP 2 manual is NOT intended to teach you the basic concepts (this is what the textbook is for!!)
  - It is NOT intended to guide you through the steps of the project
  - *It is just a description of the API to use.* The rest is for you to figure out.

What Is Described in the Chapters About Various Modules

- General introduction to the particular OS module (not a replacement for the textbook)
- The classes/methods that you are supposed to implement
- What each method is supposed to do
- Classes provided by the rest of OSP 2, which you might have to use
- Methods provided by the rest of OSP 2, which you are supposed (or allowed) to use in your implementation
- Don’t scavenge through other chapters in search of methods that (you hope) might help you to solve a particular problem!
  - Everything you need is right there in the chapter for your assigned module
  - Such scavenging won’t work, because of obfuscation, and you will be just wasting your time

Working with Git

- Basic concepts:
  - Clone (sometimes called checkout in other systems)
  - Local branch on your machine
  - Remote branch – the one on Bitbucket
  - Local branch *tracks* the remote one
    - Pull: obtain the changes made to the remote branch (by others)
      - You probably won't be in this situation unless you access the repository from different machines
    - Push: commit your changes to the remote branch, ie, save your work on Github.
      - Push regularly OR lose points!
- Use a GUI (Eclipse, SmartGit, GitKraken)

Why Github Classroom?

- Provides free private repositories with easy access by instructor/TAs
- Groups them in classrooms
- Each part of the project will be in a new repository
- Create a Github account right now! Don’t wait until too late.
  - You can apply for an academic license that provides private repositories, but it is not necessary: participation in a classroom will give you that for this course.
- Will loose project points (possibly all) for not following instructions.