Information Infrastructure II

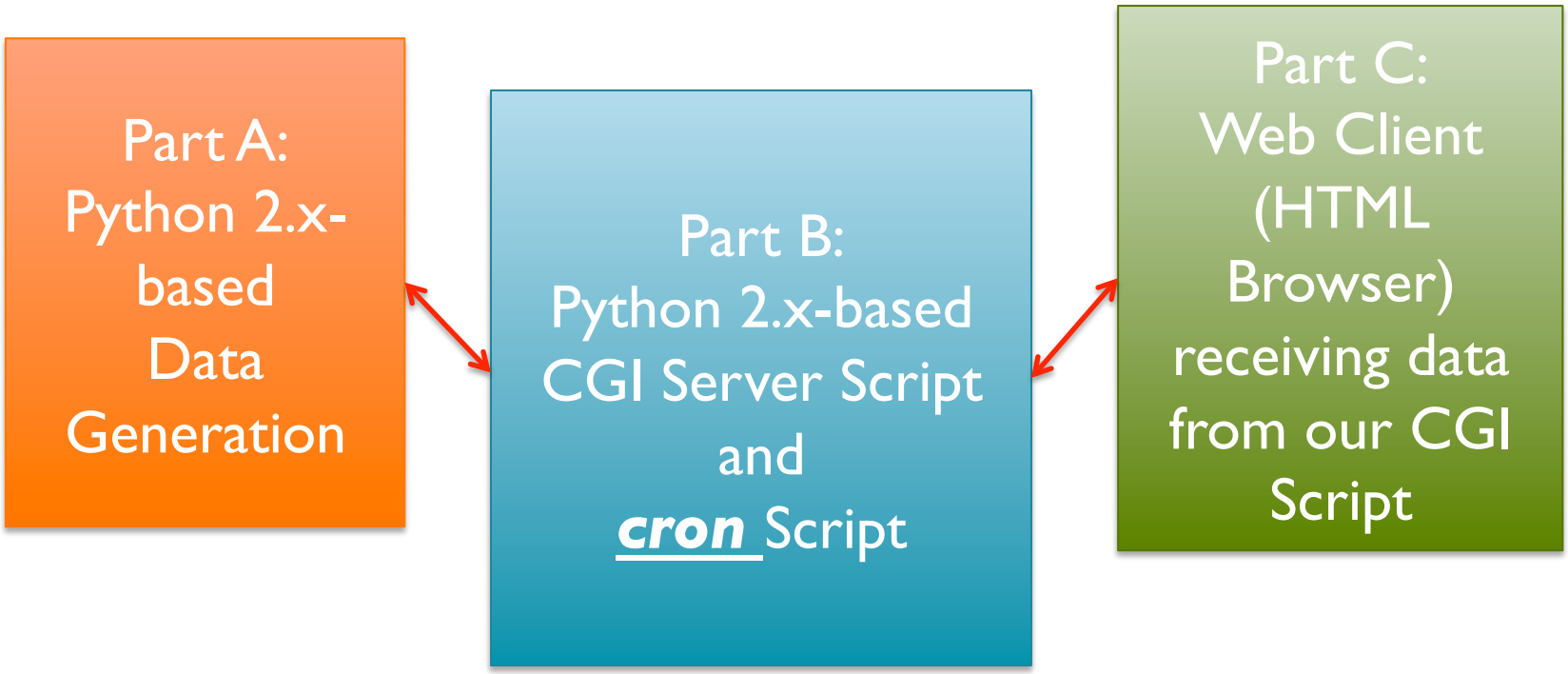
INFO I211 – Spring 2014 – Sections 18530 & 22719

Lecture 18 – 2014.03.31 & 2014.04.01

Instructor:
Mitja Hmeljak,
http://mypage.iu.edu/~mitja
mitja@indiana.edu

Assignment 1: Building a Distributed System

Assignment 1 (a1) consists of 3 parts:



Remote data generation: algorithm

- **capture image** from camera
  - (webcam, security camera, etc)
- **save image** to file
  - *(optional*: verify image EXIF metadata,
    e.g. using a Python module such as EXIF dump)
- **upload image file** to server
  - invoke *scp (*using your generated SSH key pair, we'll see that next)
- wait a bit, then "*lather, rinse, repeat*" i.e.
  - "infinite" *while* loop, with *sleep()* function between processing steps

Part A: Python 2.x-based Data Generation

Remote data generation: Python 2.x code

- upload image file to server

Part A: Python 2.x-based Data Generation

- (*optional*: naive "lock" mechanism:
  1. to avoid simultaneous read/write access to the uploaded file,
  2. set "lock file"
  3. send image file
  4. remove "lock file")

Remote data generation: how to proceed
(part A of the distributed application)

1. from Oncourse download: **a1startingcode.zip**
2. test the webcam on your laptop computer

   with the provided utility program:
   - windowssnapshot.bat (which starts CommandCam.exe) on Windows systems
   - macosxsnapshot.sh (which starts imagesnap) on Mac OS X systems
3. test the "*a1remote.py*" Python script, by running it on your laptop computer

   (you'll need Python 2.6.6 installed on it:
   from https://www.python.org/download/releases/2.6.6/ download either
   Mac Installer disk image (2.6.6) (for Mac OS X) or
   Windows x86 MSI Installer (2.6.6) (for Windows))
4. verify the *ssh* / *scp* connection from your laptop computer to silo.soic.indiana.edu
   - Windows systems need the complete PuTTY install (putty-0.63-installer.exe) from the official PuTTY site (http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html) to use *pscp*
   - Mac OS X systems can use the *scp* program that's included with the OS

Part A: Python 2.x-based Data Generation

CGI Server: Python 2.x code
(part B.2 of the distributed application)

- CGI script to collect uploaded images

- where are image files located?
  - the image file should be
    stored in a directory not accessible
    by CGI scripts
  - we'll pick ~/tmp/

- how are image files named?
  - if we upload 1 image at the time...?
  - ... and are they all called "snapshot.jpg" ?

Part B.2:
Python 2.x-based
CGI Server Script

CGI Server: running Python 2.x-based *cron* scripts
(part B.1 of the distributed application)

- image files that are automatically uploaded should be:
  - checked for integrity
  - time-stamped (rename filename)
  - made accessible to CGI scripts

- when are this tasks performed, and by whom?

- the ***cron*** *Unix system* utility!
  - We'll use it to invoke our Python script that performs the tasks above.

Part B.1:
Python 2.x-based
cron Script

CGI Server: configuring *crontab*
(part B.1 of the distributed application)

1. read the *crontab.pdf* instructions file

2. copy timestampfile.py to ~/bin/movefile.py, and **edit** it!

3. set a crontab file for your account on silo.soic.indiana.edu following the *crontab.pdf* instructions file

Part B.1:
Python 2.x-based
cron Script

web client-side output

- the web browser should receive an HTML page containing the image(s)
  - as collected by the server
  - from the remote data generation system

Part B.2 → Part C Web Client (HTML Browser)

- CGI script:
  - parse all image files
  - (optional) extract metadata
  - prepare HTML table with images
    - compute table dimensions (how many rows? how many columns?)
    - resize images

web client-side output
(Part B.2 – which enables Part C – of the distributed application)

1. start with the provided *index.cgi*

2. create the A1 directory structure for CGI on *silo.soic.indiana.edu*:
   ~/cgi-pub/i211/a1/index.cgi
   ~/cgi-pub/i211/a1/data/

3. modify *index.cgi* to display *all* stored images (from newest to oldest, or from oldest to newest) in properly formatted table, resize images using the <img src="..." width=""> tag.

Part B.2 → Part C Web Client (HTML Browser)