2009. 1. 9

the first code I want to write will open up the auth.log file, crawl backwards through it, and give me ~~sets of~~ the logins/ logouts I want.

Here's how it should go:

- open file
- start at last line.
- does the last line have the right structure?
- no → skip to the next
- yes → record

At this point i'll have a list of login/logout events which I can further slice and dice.

In fact, I can probably start at the top of the file and push each log in/out event onto a stack of events. Then I can pop events off of the stack and ~~line things~~ match up the login/logout events by their (i'm guessing) PID.

```
logfile= open('auth.log', 'r')
logfile.readline()      ← pops the current line off of the
                          stack
```

1

Jan 5 09:49:57 hermes kdm: :0[5703]: pam_unix(kdm:session):
session opened for user jrsmith3 by (uid=0)

There is a lot of extra stuff in these logfiles that I just don't care about. I can pretty much search the files for the words "kdm" and "jrsmith3". Lines with both words are all the lines I care about.

```
import re
        logfile = open('auth.log', 'r')
        curline = logfile.readline()
        if re.search(pattern, curline):
                          ↑
                    reg. exp containing kdm and jrsmith3
            # push curline onto the stack
```

At this point I should have all of the lines I want on the stack in order.

Basically I want to match the pattern $\overset{\text{<some stuff>}}{\smile}$ kdm <some stuff> jrsmith3 $\overset{\text{<some stuff>}}{\smile}$

Its possible I wont easily be able to ~~to~~ that. Probably I should try

```
        if re.search('\bkdm\:\b', curline) and \
           re.search('\bjrsmith3\b', curline):
```

Ryan
602 885 0331

2009.1.10

Once all of the relevant lines are in a list, I need to still do the following:

- match up logins|logouts
- convert the date part of the strings to a datetime object
- convert the pairs of datetimes to the start and end times of ics events
- generate an ics file from the ics events.

There are many problems that I havent addressed here. What about dangling logins (a login/logout that doesn't have a match)? The logfiles only go back so far. Am I just going to have some sliding time window for my ics file? I'd prefer to catch all of my login/logout data over time.

2009.1.26

im trying to work on this log parser. I need to open a file and print every line, line by line

I now have code that opens a logfile, parses the data to see when ive logged in and out of kdm, then generates a list (a stack) with those entries. im using a stack pattern with list.append() and list.pop() as a last in, first out thing. The new items are pushed and popped off the back of the list.

whats the next order of business? I need to take the list generated by my code and distill it into a form that can be used to create ical events. There are also features I need to implement like eliminating dangling login/logout pairs.

I also need to write some unit tests for my code.

Features:

- grab the date & time from an entry and turn it into something useful (datetime object?) noting that there is no year date in the login/out entries.
- match a login to a corresponding logout.
- what if the computer loses power while logged into kdm?
- create ical events from login/logout pairs.
- avoid clobbering or rewriting events to the target ical file
- what if the corresponding login/logout is in another log file? (an archived auth.log.1.gz, e.g.?)

I might consider setting up an svn repo and a trac server.

Some items:
- set up svn repo for this project. host it on farnsworth
- write unit tests
- set up trac to deal w/ bugs & features

Is this project object oriented or procedural? How can I cast it as oop?

2009.2.9

im working on this log parser some more. Right now I should mention some items: im writing this program not knowing exactly what is dumping output to auth.log nor knowing exactly what the syntax is: im more or less inferring a pattern. The point is that there are a host of exceptions that could exist, but don't necessarily exist or may never exist.

I plan on using the string split() method and some logic to construct my login/logout events. Regular expressions are a way fancier way to do it with more difficulty and more bugs.

these are the exceptions I have to watch out for:

~~login/logout event spans more than one auth.log file~~

- dangling login ~~logout~~ - user still logged in while script runs
- no year given inside log files
- new year's problem: a file dated the current year may have entries from the previous year.
- dangling logout: login event in previous file.

these are the important elements in the list of a single logfile line:

- elements 0,1,2 : month, day, time of entry
- " 5 : contains the (im assuming) process id for the login/out attempt. must be matched to [illegible] [illegible] its mate
- " 8 : the open or close of a session. Probably using this datum makes the system overdetermined.

1

By making several assumptions, I can come up with some logic for constructing events. Lets assume there are no interleaving kdm logins ⇒ login/logout events sit side by side in the logstack list.

First, pop the last item off the list. If the 8th element is "opened" then the user is still logged in, go to the next item. If not, grab the date and time and add the year. Pop the next item off the list. Check to see 8th element is "closed". Generate date & time from the data. Check item 5 against the previous line's item 5. Match → generate event from the pertinant date and time. Continue until the end of the list is reached. Handle dangling logout properly (throw it out).

I'll also have to deal with multiple auth.log files. The trick here is to do the same regular expression stuff that im doing with a single file in parser.py, but add code that walks backwards through the oldest file, stacking up the kdm login/outs from oldest (lowest index) to newest (highest index). Once that stack has been built, the program flow from the prev. paragraph still stands.

Assumptions:-

- no interleaved kdm logins
- nobody except the daemon writing to the logfile has altered the file
- nobody has altered the date on the file.
- nobody has altered the filenames
- the clock on the system is reasonably accurate.
- I login and out of my computer more or less every day.

Once the ics events have been generated, I want to push them into a calendar file that may contain some duplicates. Lets assume I have a stack of ics events with the most recent date at the bottom (pop) gives the newest event on the stack. I guess the idea would be to pop items off of the parsed events stack and stop popping once I reach an event that coincides with the newest event in the existing calendar. once that happens or I reach the end of the stack, I push all of the popped events into the ics file and save it.

I should have all of the tools if I get the following ~~[illegible]~~ ubuntu package:

python-vobject

which pulls in python-dateutil; a package that lets me create a datetime object from the data im pulling out of the logfile.

The last thing to do is to try to figure out how to solve the new year's problem. what would happen if I just ~~left the~~ added the year from the file?

dec	31	09:00:00	opened
dec	31	17:00:00	closed

most likely, the login/logout would be on the same day and all I'd wind up with is an event at the end of the current year. The other option, assuming I logged in on Dec 31 (or earlier), and out on Jan 1 (or later) would be an event that basically spanned the entire year.

The solution is to examine the dates on the files and the month values in the stack.

Dec	29
Dec	30
Dec	31
Jan	1
Jan	2

~~Ide~~ If I see a stack like that, im spanning the new year. This next idea conflicts with a previous idea: open ~~to~~ auth.log. Grab kdm events. If stack doesn't match above pattern, stick file year onto dates. Otherwise add file year to most recent (Jan) and file year-1 to less recent (Dec). Once years have been added, cat lists from various auth.log files together.

Note: it looks like syslogd writes to the log files.

it occurs to me that I could avoid all of this log parsing business by writing a script that runs each time I login (bash can do this, and i'm sure there's a way to do this in kde). This script records the date and time of login and the date & time of logout (when I logout).

allegedly: ~/.kde/Autostart

~/.kde/shutdown

Put scripts in there, grab current date & time, write to ~~[illegible]~~ an ics file.

Actions:

- figure out python-vobject
- ~~[illegible]~~ source folder to svn

8

2009.2.12

After learning more here's how my program is going to work:

- upon login, a script in ~/.kde/Autostart records the date: date > /home/jrsmith3/.../somefile.txt
- upon logout, the date is determined. Then the script grabs the info in the somefile.txt file and creates an ics event from that info. The resulting event is ~~[illegible]~~ inserted in an ics file somewhere. Logout script lives in ~/.kde/shutdown

My events need the following:

dtstamp
dtstart
dtend
summary
time zone

Here's how it works:

Login script runs and creates an ics event with dtstamp, dtstart, summary = At work, and timezone. Maybe also my email address, saves this half of the event in an appropriate location (see kde fhs).
Logout script runs and opens previously saved event. Adds dtend field. Deletes temporarily saved login half of ics file. Adds new event to a specified icalendar.

1