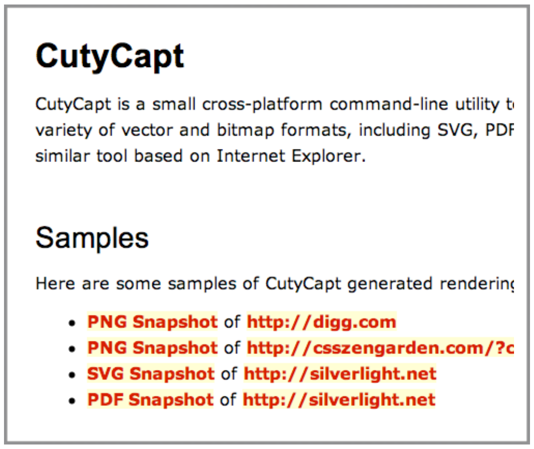
Documentation for HEPA Screensaver:

This documentation describes the steps taken to create the screensaver for the Florida Tech High Energy Physics Lab computers.



CutyCapt

CutyCapt is a small cross-platform command-line utility t
variety of vector and bitmap formats, including SVG, PDF
similar tool based on Internet Explorer.

Samples

Here are some samples of CutyCapt generated renderin

- **PNG Snapshot** of **http://digg.com**
- **PNG Snapshot** of **http://csszengarden.com/?c**
- **SVG Snapshot** of **http://silverlight.net**
- **PDF Snapshot** of **http://silverlight.net**

The screensaver shows images captured from various URLs. Since there is a built-in screensaver for Linux that shows all pictures in the "Pictures" folder, all that was needed was a program that would take screen captures of the websites and store them into the Pictures folder automatically every few minutes.

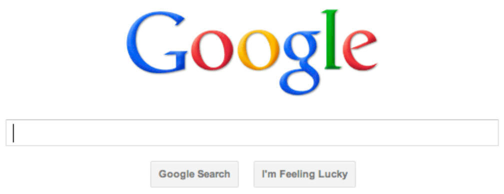
In this implementation, the free command line utility "CutyCapt" was used to take the screen captures, a bash script was written to combine all of the necessary captures, and crontab was used to run the script at regular intervals.

1. Install cutycapt at http://cutycapt.sourceforge.net/

- Recommended to add directory to path: `PATH=<cutycaptdirectory>:$PATH`
- To test, try the command:

```
xvfb-run --server-args="-screen 0, 1024x768x24" /home/gridops/
Pictures/cutycapt/CutyCapt/CutyCapt --url=http://www.google.com
--out=/home/gridops/Pictures/test.bmp
```

- The result should be a new image in the gridops Picture directory called 'test.bmp' that looks somewhat like the following:



- This test command assumes that the CutyCapt program is contained in /home/gridops/Pictures/cutycapt/CutyCapt. If it isn't, substitute the directory where CutyCapt is located.
- Note that it doesn't work if only "google.com" is entered as the URL.
- If the CutyCapt directory was added to the path, you should be able to get away with "`CutyCapt --url...`," but it is better to make sure the full path works because that is what will be used in crontab.

- If this test screen capture does not work, check the environment and path.

2. Create a script called "snapshot.sh"

- Contents of the script:

```
#!/bin/sh -v
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/
bin:/home/gridops/Pictures/cutycapt/CutyCapt

#OSG
xvfb-run --server-args="-screen 0, 1024x768x24" /home/gridops/
Pictures/cutycapt/CutyCapt/CutyCapt --url=http://
myosg.grid.iu.edu/map?all_sites=on --zoom-factor=0.80 --min-
width=1024 --min-height=768 --out=/home/gridops/Pictures/
osgsites.bmp

#DIAGNOSTICS
xvfb-run --server-args="-screen 0, 1024x768x24" /home/gridops/
Pictures/cutycapt/CutyCapt/CutyCapt --url=http://uscms1.fltech-
grid3.fit.edu/diagnostics.php --zoom-factor=0.80 --min-
width=1440 --min-height=900 --out=/home/gridops/Pictures/
diagnostics.bmp
xvfb-run --server-args="-screen 0, 1024x768x24" /home/gridops/
Pictures/cutycapt/CutyCapt/CutyCapt --url=http://uscms1.fltech-
grid3.fit.edu/diagnostics.php --zoom-factor=0.80 --min-
width=1440 --min-height=900 --out=/home/gridops/Pictures/
diagnostics.bmp
#GANGLIA
xvfb-run --server-args="-screen 0, 1024x768x24" /home/gridops/
Pictures/cutycapt/CutyCapt/CutyCapt --url=http://uscms1.fltech-
grid3.fit.edu/ganglia --zoom-factor=0.80 --min-width=1440 --min-
height=900 --out=/home/gridops/Pictures/ganglia.bmp
xvfb-run --server-args="-screen 0, 1024x768x24" /home/gridops/
Pictures/cutycapt/CutyCapt/CutyCapt --url=http://uscms1.fltech-
grid3.fit.edu/ganglia --zoom-factor=0.80 --min-width=1440 --min-
height=900 --out=/home/gridops/Pictures/ganglia.bmp

echo $(date) >> /home/gridops/cron_screensaver.log
echo $PATH >> /home/gridops/cron_screensaver.log
```

- Doing the capture twice ensures that it works.
- The settings used in the CutyCapt commands were found through experimentation.
- The #! statement (known as the shebang or hash-bang statement) at the top indicates what shell to use when running the script (in this case, sh as opposed to maybe csh or ksh).
- The '-v' at the top puts the script into verbose mode, meaning that it will echo the status after every command. You can do ‘-vx’ for even more echoing or just ‘-x’ if ‘-v’ isn’t good enough.

- At the end, the script echos the date and path to a log file for debugging purposes.
- This script can probably be vastly improved by creating an array of URLs to capture and using a foreach loop to run the command with each URL.
- Attempts were made to shorten the program, but for some reason CutyCapt crashes and burns unless it's done the long way.
- Make sure the script permissions are correct. You can fix them with:

```
chmod +rw snapshot.sh
```

- To run the script, use:

```
./snapshot.sh
```

- It is a good idea to make sure that the script works manually before putting it in crontab.

3. Open up the crontab with `crontab -e`. If opened for the first time, select an editor.

- When it asks you to select an editor, it will recommend nano as the easiest one to use. Just remember that exiting and saving is Ctrl + x.
- Be careful not to accidentally do "crontab -r" (the 'e' and 'r' keys are right next to each other, so it happens to some people). That *removes* the crontab and is *not* fun to fix.
- Tell it to run the snapshot script:

```
* * * * * /home/gridops/snapshot.sh 2>> /home/gridops/
cron_screensaver.log
```

- This runs the script once every minute.
- The part with `2>>` means that error output from the script will go to a log file in the gridops home folder. This is for debugging purposes.
- Note: crontab has a very limited environment, so it is likely to break unless full paths are specified.

Diagnostics Site Documentation

Plain english documentation for diagnostics website
Website is located at http://www.uscms1.fltech-grid3.fit.edu/diagnostics.php

Basic Info

- The diagnostics page uses a combination of html, php, and css. The necessary files are:

diagnostics.php	- Contains the php and html code that *generates* the html
style.css	- Contains the css for *formatting* the html

- These files are located at /var/www/html.
- Will probably need sudo privileges to edit.
- All of the coding was done in the vi editor.
- The diagnostics.php file only manages the *content*, which includes the SAM tests, Ganglia graphs, toolbar, general information (hosts up/hosts down, etc.) and logos.
- The style.css file manages the *way the content is formatted*, such as the font colors, background images, image sizes, and navigation bar colors.

Features

- Takes arguments from url to decide which ganglia page to load
- Graphs are organized by node
- Graphs are clickable, show a zoomed version or more details
- Background stretches to match window size
- Logos are clickable and direct to respective websites
- Refreshes every five minutes

Sources for PHP, CSS, and HTML help

http://www.php.net/manual/en/index.php - Contains documentation for php functions
http://www.w3schools.com - Contains tutorials on PHP, CSS, HTML, etc.

Comments

```
// commented out line in PHP
/*
        commented out block in PHP
*/
<!-- Commented block in HTML -->
```

- The PHP code should be fairly well commented, so you may find out more by going there first.

Indicating where PHP code should go in an HTML file

```
<?php
        "Bubble" of PHP Code
?>
```

or

```
<?php “Bubble" of PHP Code ?>
```

Getting URLS

- The PHP functions file_get_contents() returns a string containing the html source code of the webpage.
- Diagnostics website stores the URL of the SAM Tests table into a string variable, and then stores the contents of that page into another variable called $sam.

Deciding which Ganglia page to load

- The page uses the special variable GET to get the arguments in the url. These are the words following .php?.
- Getting the $ganglia variable involves this statement:

```
isset($_GET[time]) && $_GET[time] != ""
```

- Here the program is checking to see if the variable "time," set by the URL, has a value and if it isn't a blank string. The function 'isset' first ensures that _GET has a value.
- For example, adding "?time=string" to the end of the URL will declare the variable "time" and set it to "string."
- The rest is just a series of if statements checking if time contains day, month, or year and sets the URL to the appropriate string. If time is not equal to any of those strings, then it uses the default ganglia page, which is organized by load rather than node.

Formatting the HTML from file_get_contents

- Since the SAM test table was designed for a white background and the diagnostics page has a black background, some of the colors had to be swapped.
- It was a little messy, but the editing was done by using PHP's "str_replace" command.
- str_replace takes three arguments. The string to search for, the string to replace it with, and the string to be searched.
- Rather than nest a bunch of str_replace statements, the variables were modified over several lines.
- Nothing happens if the string in question cannot be found.

What was changed in $SAM

- Replace gray background colors with black
- Shorten the string "uscms1.fltech…" with "CE" and "SE" for better formatting
- Center and correct text colors in table

Printing the SAM tests and logos

- The SAM tests are actually in a 1x5 table with the logos on either side
- The table was created with HTML:

```
<table width=100% border="0">
```

- This command creates a table that stretches with the window and has no borders

Printing an error statement when there are errors

- The program "detects" SAM test errors by using the strpos() function.

- When the word "error" turns up in $sam, strpos("error", $sam) will return a positive number.
- If the word error is not found, the function returns a -1.
- So if the position is a positive integer, the program echos a red error message onto the screen.
- The PHP command "echo" will echo text to the html file, so echoing html formatting commands such as <br> and <font color = "red"> will format the text.
- However, escape sequences are needed for quotes because the entire line must be surrounded by quotes.

Getting text from the Ganglia page

- The text sequence from ganglia containing the number of CPUS, hosts, etc. is exactly 381 characters long in the source code.
- To "grab" this data, a type of "find and grab" algorithm (not an actual algorithm, just my description for it) was used that goes like this:
    1. Use strpos to find the start position to grab from.
    2. Use substr to grab the string from that location with a certain length.
- substr has three arguments: the string to get data from, an integer index to start at, and a length of a string to grab.
- This text was formatted to add spaces and breaks.

The Pie Graph

- There is also a pie graph on the Ganglia Page.
- There really wasn't a good place for it, and the method to display it never worked.
- The code is still there, but it has been commented out.

Overview graph information on Ganglia Page

- There are four overview graphs on the ganglia page.
- The code uses a combination of strpos and substr.
- First, strpos is used to find the position of the image url (knowing that the url begins and ends with a certain unique string of characters). The length of the url is also calculated.
- Then, substr is used to extract the url of that image and put it into the html.
- This works because strpos has an optional integer argument that tells the function where to start searching.
- Therefore, if the end of the previous URL is set as the optional argument, strpos will find the position of the next graph.
- This method utilizes the fact that the strpos command can take a third argument, which is an index to start searching at.
- So in summary, the steps are:
    1. Find the start position of the graph.
    2. Find the end position of the graph.
    3. Use the start and end positions to calculate a string length.
    4. Get the substring with that start position and length.
    5. Echo the substring.

6. Set the new "start" index to start searching at to the end of the previous substring.
7. Repeat four times to get all four graphs.

Node graph information on Ganglia Page

- A similar "find and grab" method was used for the nodes.
- The number of working nodes was grabbed from the general information substring.
- The searching algorithm was repeated for each node.
- Also in the for loop is an if statement that starts a new line after every five graphs.

Important PHP Functions Used

file_get_contents
string file_get_contents (string $filename [optional args])
When given a url or filename, returns the contents of that file as a string variable. Used to get the contents of the ganglia and sam pages.

strpos
int strpos(string $haystack, mixed $needle [, int $offset = 0])
When given a string and a substring, returns the position of that substring within the string or negative one if it does not contain the substring. Used to find the positions of certain strings so that they may be extracted with substr.
An optional argument 'offset' will tell the function where to begin looking in the string. The default value is 0 (start from the beginning).

str_replace
mixed str_replace(mixed $search, mixed $replace, mixed $subject [optional args])
When given a string and two substrings, replaces all instances of the first substring with the second substring. Used in this program to edit the contents of the text from using file get contents.

substr
string substr(string $string, int $start [, int $length])
When given a string and two integers, returns the substring from the first integer as a starting index to the second integer as the length.

(condition) ? value1 : value2 (Inline conditional)
Not actually a function, but used several times in the program. Has the value of value1 if condition is true and value2 if it is false.

Tips for Debugging

- View the page on multiple browsers: Firefox, Safari, IE, Chrome, etc.
- On most browsers, you can right-click on the page and select “View Source” to look at the HTML that the PHP code generated.

- Chrome has a neat “Inspect Element” option as well.
- If the entire page disappears, that means that the PHP has broken completely. There might be an unbalanced { somewhere.
- You can insert an `echo “Hello world”` statement into the PHP to see if the script ever reaches a certain point.