OpSim Tucson's Multi-user Environment

This is a description of the multi-user production environment that OpSim Tucson uses to set up and run simulated surveys with the Operations Simulator.

Development machines are generally single-user environments and used by an individual programmer. After a consolidation of hardware (see Archive pag e), the team's production machines consist of a MacOS machine, opsimblitz1, and a Linux machine, opsimblitz2. On the older machines, a single username was used to by all users to log on, set up and run simulations. This was simple to maintain as the machine appears to be a single-user machine even though multiple users are accessing it. With the advent of new cyber-security policies, it became necessary for each user to have their own login information and user account.

Each user has a "home" directory that is not visible to the other users, and while this ensures privacy, it is not conducive to collaboration. The simulator codebase and the analysis tools are installed and run from directories in /lsst/*. Because the set up configuration files, as well as the output files are created in this directory structure, and all users are logged in under their own user name, it is necessary to have a permissions environment such that all users can read and write (or overwrite) files in this directory structure. This also necessitates some conventions for uses of the machine so users do not overwrite each other's work, such as, verbal agreement is needed before a new user takes over use of a machine. Lastly, because output files from the analysis codebase are copied to a distribution repository on a machine called opsimcvs, permissions need to allow this activity as well as allowing overwrite privileges on the opsimcvs machine.

The new analysis codebase, MAF, while installed in the /lsst area, has the capability of running from any directory and outputting files to a specified directory. New conventions on how to store these files where they can be easily shared within the team as well as with the Science Community (selected files) will need to be developed.

Here is documentation from Iain on changes that were needed to opsimblitz1 and opsimblitz2 to set up the functionality described above.

Email to lsst-opsim-dev on Mar 20, 2014 regarding opsimblitz2

Cathy and I are happy to report that tentatively it appears the umask and permission problems with opsimblitz2 and scp to opsimcvs as users petry and lsst has been resolved!

I won't even guess how things became the way they are.

What/How was it fixed?

First we need to know what is not correct so I wrote a simple script to loop through opsim_3 sstar_4 checker_3.9.2 Prod_2.6.1 from /lsst. There are actually two scripts since I was lazy on optimization and compactness. The other script is the same as the attached script except I removed the "| wc -l" from the find -exec command.

For each directory: count of the number of directories; how many have group ownership of lsst; Not group ownership of lsst; how many with 775 permissions; Not 775 permissions; RWX group permission regardless of user and other; Not RWX group permission regardless of user and other; how many files; with 664 permissions; NOT with 664 permissions; how many are py files.

Also then looks for .cshrc files in home directories that contain umask 0002.

Also how many files are w/o extensions on the assumption they are binary files and how many are .sh files.

We searched from binary, py and sh files since using 664 made them not executables.

Actions taken:

> Change all files NOT 664 to 664;
>
> Change all files NOT group lsst to group lsst; Changed all py files to 775; In sstar/script & checker_3.9.2/utils, changed all files w/o extension to 775; Changed all .sh to 775
>
> Of course, we eyeballed the files before making changes.

Other fixes done:

> Petry key wasn't available causing scp to fail- RESOLVED.
>
> Changed output/design from schandra owner to lsst owner.
>
> Changed output directory from 770 to 775.
>
> Fixed opsimcvs passwd file so that user lsst is NOT a member of group root and is now member of group lsst (not sure why is was like that).

I am taking a look at opsimblitz1 and will implement the changes and then ask Cathy to run through the tests exactly the same way.

Iain

Email to lsst-opsim-dev on Mar 20, 2014 regarding opsimblitz2

What I have done:

Petry, lsst, kcook are all using umask 0002.

Cd /lsst

find opsim_3 sstar_4 checker_3.9.2 Prod_2.6.1 -type f ! -perm 664 | wc -l

~2802 with wrong permissions (some might be py, binaries, and sh files which will be fixed in later commands)

Now running

find opsim_3 sstar_4 checker_3.9.2 Prod_2.6.1 -type f ! -perm 664 -exec sudo chmod 664 {} \;

find opsim_3 sstar_4 checker_3.9.2 Prod_2.6.1 ! -group lsst -type f -exec ls -l {} \; | wc -l

0 which is what we want

cd /lsst/sstar/script

find . -maxdepth 1 ! -name "*.*" -type f -exec ls -l {} \;

cd /lsst/sstar/script

find . -maxdepth 1 ! -name "*.*" -type f -exec ls -l {} \; find . -maxdepth 1 ! -name "*.*" -type f -exec sudo chmod 775 {} \; cd /lsst/checker_3.9.2/utils find . -maxdepth 1 ! -name "*.*" -type f -exec ls -l {} \; find . -maxdepth 1 ! -name "*.*" -type f -exec sudo chmod 775 {} \; find . -maxdepth 1 ! -name "*.*" -type f -exec ls -l {} \; find . -maxdepth 1 -name "*.sh" -type f -exec ls -l {} \; find . -maxdepth 1 -name "*.sh" -type f -exec sudo chmod 775 {} \; find . -maxdepth 1 -name "*.sh" -type f -exec ls -l {} \;