Subject: creating temporary files safely
From: Joseph Picone <joseph.picone@gmail.com>
Date: 6/7/23, 9:06 AM
To: nedc_research <nedc_research@googlegroups.com>
CC: ECE 1111 <temple_engineering_ece1111@googlegroups.com>

Very often when writing a script, you have a need to create an intermediate file. This is not as easy as it might sound, and there are fairly standard ways to do this.

Suppose you adopt a simple approach:

```
myscript.sh > temp.dat
another_script.sh temp.dat
```

What is wrong with this approach?

With a little thought, you realize that only one instance of this program can run at a time. So if you run two versions of this simultaneously, there will be problems.

Further, the location of the file might not be a place that all users an write to (e.g., /home/picone/rje/x.dat).

A common solution in Linux is to write a file into /tmp. In fact, in the ISIP env, we have defined a variable (ISIP_TMPDIR=/tmp) that you should use to write a temp file.

But is this safe?

Not really, because it doesn't allow you to run multiple instances of a program... unless the filename you create is unique. And... you must remember to clean up /tmp when the program exits or crashes - not an easy thing to do. And... there are race conditions - two processes running on different machines accessing the same location can cause problems. It is not necessarily what we call network safe.

Unix offers an interesting solution: 'man mktemp'. This is a function/command that will generate a unique filename each time you call it. This is actually a pretty good solution. But it is still not 100% safe, especially if there are multiple machines accessing the same directory. This is why sometimes we will use add the process id and the time of day to the filename. This reduces the chance that two processes can use the same filename.

In the original ISIP env, we have a C++ function that handles all this. Python has a tempfile library that also does these things.

Just remember when you create such a file, you need to delete it before the program crashes or exits. So error handling is essential. Otherwise, you end up littering your filesystem with tons of useless files, and that is not a good thing.

-Joe