IT341 Introduction to System Administration

Project VII - Using `rdist` to Distribute Files

Now that you can get to other machines without supplying a password or pass phrase each time, we can set about automatically distributing files to the clients. This is something we would want to do in an industrial-strength network. There are two tools that are useful to this task: `rdist` and `sed`. `rdist` is used to distribute files; `sed` is used to modify them slightly to accommodate the specifics of various hosts on the network. We'll look at `rdist` here and `sed` in Assignments 4 and 5.

`rdist` stands for "remote distribution". It is used to distribute files from one host to others. The idea is, that when one wants to maintain files that are to be identical on many hosts, one maintains them on one host – making modifications to files only on that host – and then uses `rdist` to distribute them to the other hosts. If the files on the various machines get out of sync, you just run `rdist`.

One might ask, what's the difference between `rdist` and `rsync`? The answer is: it's a matter of purpose.

- `rdist` is for *distributing* files on the network
- `rsync` is for *backing up (and restoring)* file systems. We will explore `rsync` in Project 8.

Each has behaviors particular to its purpose. What might be a scenario where you would want to use `rdist`? Where you would want to instead use `rsync`? Write about this in your admin log.

Installing `rdist`:

The first thing we have to do is install `rdist`, on our host; I've installed it on `it20` but you can install yours on your virtual machine server/client[1]. Assuming you are logged in as `sysadmin`...

(Do this):

```
sudo apt-get update[2]
sudo apt-get install rdist
```

Now log out and log in as yourself. We are taking advantage of the fact that we set up key authentication in the last project.

NOTE: *You <u>cannot</u> complete this unless you have successfully implemented **<u>key-based authentication</u>** from **<u>project 6</u>**!*

[1] In practice, one chooses a single host, usually a server from which to distribute files but we are just playing here – so play away.

[2] When you do this it is a good idea to install any updates available.

Configuring `rdist`:

(You will do this part by creating myrdist, making it executable, and creating Distfile)

(If you are not familiar with scripting in a Linux-based environment, consider brushing up on this topic. Google phrases like **linux shell script tutorial**)

By default, **`rdist`** uses the (non-secure) `rsh` for transport. We want to use the more secure **ssh**. So we create our own script **`myrdist`** – which invokes **`rdist`** with the proper parameters:

```
#!/bin/sh
# A preconfigured rdist that uses ssh
SSH="`which ssh`"
RDISTD="`which rdistd`"

rdist -p "$RDISTD" -P "$SSH" "$@"
```

Of course, we must insure that this file is *executable*:

```
chmod +x myrdist
```

Then, we must define a *distfile* (the default name is **`Distfile`**), which describes the sorts of distributions we might want to do. A *distfile* is similar to `make`'s *makefile*; it provides for a list of target tasks, and specifies how each task is to be carried out. For example, consider where we want to copy `/etc/hosts` from `it20` (or your client) to **`itvm28-1b`**; the **`hosts:`** label specifies the target and what follows describes what must be done:

```
# Distfile for distributing files

hosts: /etc/hosts -> ( itvm28-1b )
install /etc/hosts ;
```

This specifies that to satisfy the target `hosts`, we copy file `/etc/hosts` to **`itvm28-1b`**, and install it as `/etc/hosts` there. (**Do Not Do This!!!**)

Be extremely careful; a wrong `Distfile` can cause havoc! See below.

Running `rdist`

To run **`rdist`**, we simply type **`rdist`**, or to use our configured version, **`myrdist`**. But before doing so, we can modify our **`Distfile`** by adding a verify option to the install:

```
# Distfile for distributing files

hosts: /etc/hosts -> ( itvm28-1b )
```

```
install -overify /etc/hosts ;
```

This says what **`rdist`** ***would*** do in the current environment, but it doesn't actually do it. **`rdist`** generally only overwrites files whose modify dates are older than the files being copied, although one may change this behavior using options. See the **`rdist`** man page.

What are some of the most important options available for **`rdist`**? Why is that? Write about this in your admin log.

Play in a sandbox

NOTE: The following section is not intended to be carried out by you. It is just here to serve as an ***example***.

Before using **`rdist`** to distribute files to an important directory such as `/etc`, set up a sandbox and practice there. For example, use `/tmp` at the destination end:

```
# Distfile for distributing files

hosts: /etc/hosts -> ( itvm28-1b )
install /tmp/hosts ;
```

To execute **`rdist`**, we simply type

```
rdist
```

Be careful because **`rdist`** will overwrite a directory with a file. Saying

```
hosts: /etc/hosts -> ( itvm28-1b )
install /etc/hosts ;
```

is fine as you specify a file name as the destination; **`rdist`** will write a new `hosts` file in `/etc`. However…

```
hosts: /etc/hosts -> ( itvm28-1b )
install /etc ;
```

is disastrous. It will overwrite the `/etc` directory with the file hosts, removing `/etc` altogether. (It has happened to previous instructors!)

On the other hand, if you are distributing several files, as in…

```
hosts: ( /etc/hosts /etc/nsswitch.conf) -> ( itvm28-1b )
install /etc ;
```

…then the behavior is as you would expect: the two files hosts and nsswitch.conf are copied into directory /etc on **itvm28-1b**.

What precautions can you take in order to keep yourself from making such? Write about this in your admin log.

Also, you can ask that one directory overwrite another directory. The following will overwrite /etc on **itvm28-1b** with the /etc on the machine running **rdist**

```
hosts: /etc -> ( itvm28-1b )
install /etc ;
```

An Exercise (Do this)

As an exercise, on your client, copy /etc to /tmp

```
cp -r /etc /tmp/<your-login-name> # e.g. /tmp/abird
```

Then, write a Distfile that copies some of those files that we have been defining, and that we want on all clients, over to another client's (e.g. another **it2x-yz**'s) /tmp/<your-login-name>/etc

Experiment with various Distfiles. And, be careful! **Never use a target that doesn't involve /tmp**