```
$LABTAINER_DIR/labs/[labname]/[container_name]/_bin/noskip
```

Files whose basenames match any found in `noskip` will be collected.

Alternately, a file at /var/tmp/home.tar will be expanded into the user home directory. Use the Docker `COPY` directive to place a file here. See the

```
$LABTAINER_DIR/scripts/designer/base_dockerfiles/Dockerfile.labtainer.firefox
```

for an example. These files will not be collected unless they are newer than the original file, or if the base file name appears in the `noskip` list described above.

4.4 Lab-specific system files

All files in the

```
$LABTAINER_DIR/labs/[labname]/[container name]/_system
```

directory will be copied to their corresponding paths relative to the root directory. For example, configuration files for /etc should appear in _system/etc/.

The initial Dockerfile from the templates include this line:

```
ADD $labdir/sys_$lab.tar.gz /
```

to accomplish the copying. If a lab contains a large quantity of system files, or large files, those can be placed into the directory named:

```
$LABTAINER_DIR/labs/[labname]/[container name]/sys_tar
```

either as individual files, or in a "sys.tar" archive. In the former case, the framework will automatically create the sys.tar file. This technique can save time in building lab images because the files do not need to be archived for each build.

In general, files modified and maintained by the designer should go into the _system directory while static system files should go into the sys_tar directory.

NOTE: CentOS systems do not have a `/bin` directory, that is actually a link. If you create a `_system/bin` directory for the lab, that will trash the `/bin` link and result in an obscure Docker build error.

4.5 System services

The general Docker model is that a single Docker container runs a single service started via the ENTRYPOINT command, with logging being forwarded to the host. Labtainers disregards this model because our goal is to make a container look more like a Linux system rather than a conformant Docker container. Labtainer Dockerfiles for Ubuntu and Centos containers use systemd based images that run the /usr/sbin/init process. [10] The labtainer.network configuration of the baseline Dockerfile also starts xinetd, which will then fork services, e.g., the sshd, per the /etc/xinet.d/ configuration files.

Services should be added using `systemd` constructs. For those of us who often forget what those are, a simple web server service can be added to a container by unpacking this tar from the witin the container's directory:

```
tar -xf $LABTAINER_DIR/scripts/designer/services/web-server.tar
```

[10] Now deprecated Ubuntu-based Labtainer Dockerfiles included an ENTRYPOINT command that launches a *faux_init* script that starts rsyslog, (so that system logs appear in /var/log), and runs rc.local.