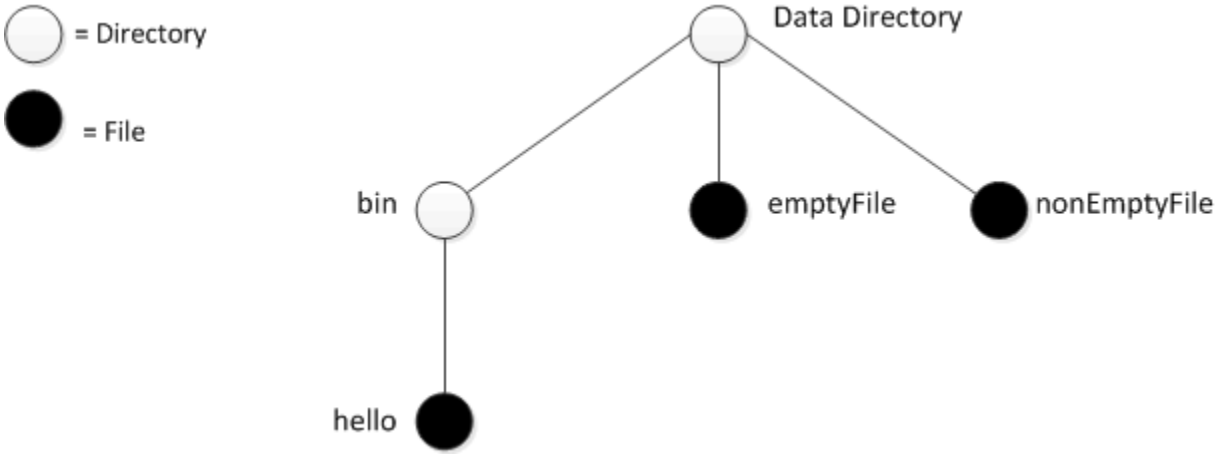
Using icepatch2calc

This page describes how to prepare a file set using `icepatch2calc`.

On this page:

Preparing a File Set using `icepatch2calc`

Suppose we have the directories and files shown below:



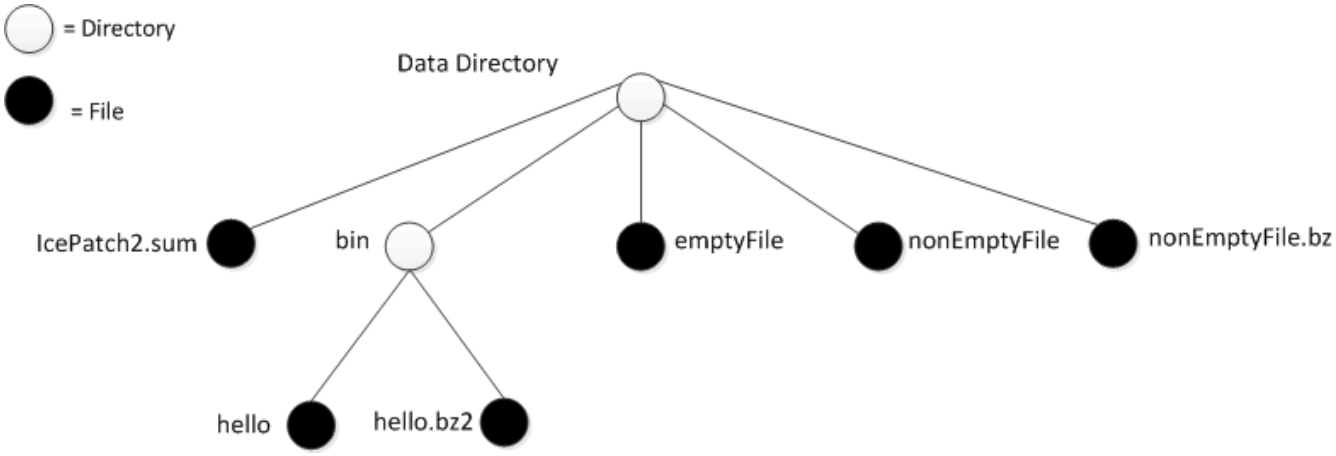
An example data directory.

Assume that the file named `emptyFile` is empty (contains zero bytes) and that the remaining files contain data.

To prepare this directory for the transmission by the server, you must first run `icepatch2calc`. (The command shown assumes that the data directory is the current directory.)

```
$ icepatch2calc .
```

After running this command, the contents of the data directory look as follows:



Contents of the data directory after running `icepatch2calc`.

Note that `icepatch2calc` compresses the files in the data directory (except for `emptyFile`, which is not compressed). Also note that `icepatch2calc` creates an additional file, `IcePatch2.sum` in the data directory. The contents of this file are as follows:

```
. 3a52ce780950d4d969792a2559cd519d7ee8c727 -1
./bin bd362140a3074eb3edb5e4657561e029092c3d91 -1
./bin/hello 77b11db586a1f20aab8553284241bb3cd532b3d5 70
./emptyFile 082c37fc2641db68d195df83844168f8a464eada 0
./nonEmptyFile aec7301c408e6ce184ae5a34e0ea46e0f0563746 72
```

Each line in the checksum file contains the name of the *uncompressed* file or directory (relative to the data directory), the checksum of the *uncompressed* file, and a byte count. For directories, the count is `-1`; for uncompressed files, the count is `0`; for compressed files, the count is the number of bytes in the *compressed* file. The lines in the file are sorted alphabetically by their pathname.

If you add files or delete files from the data directory or make changes to existing files, you must stop the server, run `icepatch2calc` again to update the `IcePatch2.sum` checksum file, and restart the server.

`icepatch2calc` Command Line Options

`icepatch2calc` has the following syntax:

```
icepatch2calc [options] data_dir [file...]
```

Normally, you will run `icepatch2calc` by simply specifying a data directory, in which case the program traverses the data directory, compresses all files, and creates an entry in the checksum file for each file and directory.

You can also nominate specific files or directories on the command line. In this case, `icepatch2calc` only compresses and calculates checksums for the specified files and directories. This is useful if you have a very large file tree and want to refresh the checksum entries for only a few selected files or directories that you have updated. (In this case, the program does not traverse the entire data directory and, therefore, will also not detect any updated, added, or deleted files, except in any of the specified directories.) Any file or directory names you specify on the command line must either be pathnames relative to the data directory or, if you use absolute pathnames, those pathnames must have the data directory as a prefix.

The command supports the following options:

- `-h, --help`
  Displays a help message.
- `-v, --version`
  Displays the version number.
- `-z, --compress`
  Normally, `icepatch2calc` scans the data directory and compresses a file only if no compressed version exists, or if the compressed version of a file has a modification time that predates that of the uncompressed version. If you specify `-z`, the tool re-scans and recompresses the entire data directory, regardless of the time stamps on files. This option is useful if you suspect that time stamps in the data directory may be incorrect.
- `-Z, --no-compress`
  This option allows you to create a client-side checksum file. Do not use this option when creating the checksum file for the server — the option is for creating a client-side `IcePatch2.sum` file for updates of software on distribution media.
- `-i, --case-insensitive`
  This option disallows file names that differ only in case. (An error message will be printed if `icepatch2calc` encounters any files that differ in case only.) This is particularly useful for Unix servers with Windows clients, since Windows folds the case of file names, and therefore such files would override each other on the Windows client.
- `-V, --verbose`
  This option prints a progress message for each file that is compressed and for each checksum that is computed.

See Also

- Running the IcePatch2 Client