Joe Werne, CoRA/NWRA
Chris Bizon CoRA/NWRA
Michael Gourlay CoRA/NWRA

GWD-I
CoRA/NWRA
September 19, 2001

# `archive`: A facility-independent tool for managing archival data.

September 19, 2001

Status of this memo: Informational

**Abstract**

U.S. national supercomputer centers present users with a plethora of command syntax, semantics, and functionality for accessing archival data storage. The DoD alone operates and maintains 21 different supercomputer centers for its 5000 users. To simplify data access for users, the DoD is sponsoring the development of a uniform command-line interface for data storage and retrieval at all of its centers. Here we report the application programming interface (API) `archive` currently being developed by the DoD so that others may learn of and contribute to the proposed syntax. The intent is to provide a facility-independent interface for the user to access archival storage data. We hope that feedback will help shape the API so that it is more broadly used and eventually becomes a standard. We anticipate many grid-related applications share identical needs with those for which `archive` is currently being developed.

## Contents

**1. The Problem: Data Management Heterogeneity.** **1**

**2. Proposed Best-Practice Solution: Facility-independent API.** **2**

**3. Author Contact Information** **3**

**4. Acknowledgments** **3**

**5. Glossary** **3**

**A. `archive` man page** **5**

## 1. The Problem: Data Management Heterogeneity.

Current large-scale supercomputer applications within the DoD duplicate on a modest scale some of the problems faced by grid technologies and related software solutions. For example, data files generated during

1000-processor computations at U.S. national supercomputer centers can fill a significant fraction of the total available disk space. Hence, automated movement of this data to archival storage systems, and sometimes off-line to tape or other media is necessary.

The tasks associated with data archiving, though straightforward, are complicated by the differing syntaxes for common tasks offered at different supercomputer centers. For instance, putting a file on the data archive may be performed through remote shell commands (rsh, rcp), through an ftp-syle interface (pftp), or through other facility-specific commands (far put, hsi put, msfput). When end users who are trained in mathematical and physical sciences (i.e., not computer programming or computer science) confront the script-programming challenges presented by the many differing supercomputer center commands, their solutions are often inefficient and cost them precious labor hours to maintain during the course of their research.

## 2. Proposed Best-Practice Solution: Facility-independent API.

Several of the DoD Challenge Users possess supercomputer accounts on multiple platforms at multiple facilities; hence, they are impacted directly by facility heterogeneity. Though a large fraction of such users continue to maintain scripts with hard-coded substitutions to accommodate differences between centers, a subset of Challenge Users have adopted the practice of writing software abstraction layers to convert general user commands into facility-dependent syntax. We have integrated what we feel are the best-practice solutions developed by these users into a portable data-management interface. Users can now write scripts that require data archiving in a facility-independent manner by simply using the `archive` command, which itself handles any facility-dependent behavior.

We are currently in the process of 1) standardizing the software solutions that have arisen naturally from these DoD users, 2) implementing a data-migration interface written in Perl, 3) soliciting feedback from users to enhance or modify the API, and 4) implementing the API for NSF or other supercomputer facilities.

`archive` is a particular application in The Practical Supercomputing Toolkit (www.pstoolkit.org). PST is an open-source project, allowing facilities and users to benefit from the experience of others. Though the DoD has pledged that all of its centers will adopt the API, individual users can download or easily implement solutions for centers that do not yet officially offer the API (because they are not one of the DoD centers and likely they do not yet know the API exists). The choice to allow users to download the software and install it in their home directory was made because users are active in the development, and they do not desire to wait for official sanctioning by a particular center before easily porting code that already uses the `archive` command interface.

The current man page for the `archive` API is included as an appendix to this document. Its functionality includes methods for adding, retrieving, and removing files to/from a data archive (e.g., a supercomputer center's mass-storage facility), listing archive files, changing a file's mode specification, and migrating data to/from slower off-line media. Solutions are currently operational at four DoD centers and one NSF Center: Army High Performance Computing Research Center (AHPCRC), U.S. Army Engineer Research Development Center (ERDC), Naval Oceanographic Office (NAVO), Aeronautical System Center (ASC), and Pittsburgh Supercomputer Center (PSC).

`archive` is written as two Perl modules which comprise distinct software layers. All of the code in the top layer (archive.pm) is facility independent. It handles command-line parsing, multiple retries, and other logic. It makes calls to the second layer (local.pm) which contains a number of very short subroutines whose implementations are facility dependent. local.pm must be implemented for each center, but its subroutines

usually invoke only a single line of executable code each. The philosophy is to simplify as much as possible implementation and maintenance of local.pm since it must be ported to individual centers. More complex logic is left to the archive.pm layer.

The implementation of local.pm is aided by the inclusion of user-submitted examples for specific centers; e.g., local.pm.ahpcrc, local.pm.erdc, local.pm.navo, local.pm.asc, and local.pm.psc. Current implementations of the local layer deal with data archives which are NFS mountable, those that use samFS commands via remote shell, and some that use commands developed at a particular facility. With these implementations already availible, it should be a simple task for administrators of a particular facility to tailor a local.pm to fit their needs by following the examples of others.

By implementing and using `archive`, abstraction of data-archive operations is achieved, and application users work more efficiently. This is our experience at the DoD centers. We anticipate it will be the case for other sites as well. Though the supercomputer application mentioned above has motivated the development of `archive`, we recognize that data movement presents a set of general problems shared by other applications. We expect that a standard API would help abstract data-migration components of many other grid-related problems.

Feedback we have receives indicates that facility to transfer data from one center's data archive to another center is desirable. We anticipate this functionality could be added to `archive` with the aid of the GridFTP protocol (www.globus.org/datagrid/gridftp.html).

## 3. Author Contact Information

Joe Werne
Colorado Research Associates Division (CoRA),
Northwest Research Associates, Inc. (NWRA),
Boulder, CO 80301
werne@co-ra.com

www.pstoolkit.org contains the man pages for both `archive` and a tool for similarly abstracting the process of job submission (`qprep`), as well as the implementation of `archive` and pointers to mailing lists for PST users and developers.

## 4. Acknowledgments

This work on PST is currently funded by the Department of Defense High Performance Computing Modernization Office. In addition to those people listed on this document, significant contributions have been made to PST by Joseph Robichaux, Ray Sheppard, and John Skinner of NAVO and David Sanders of ERDC.

## 5. Glossary

- AHPCRC: Army High Performance Computer Research Center.
- ASC: Aeronautical Systems Center.
- `archive`: A utility that provides a common command line interface for archiving data at supercomputing facilities. `archive` is part of the TUSC layer of PST.

- CoRA: Colorado Research Associates. A division of NWRA.
- data archive: A long-term storage system, usually used to hold the typically large results of computations on a supercomputer.
- DoD: U.S. Department of Defense.
- ERDC: U.S. Army Engineer Research Development Center.
- local: Related to a particular facility.
- migration: The act of moving data that is has already been archived to a still-slower form of media.
- NAVO: Naval Oceanographic Office.
- NWRA: Northwest Research Associates.
- PSC: Pittsburgh Supercomputer Center.
- PST: The Practical Supercomputing Toolkit. A collections of utilities that allow users to more productively manage supercomputing resources.
- TUSC: Tools for Uniform Supercomputing. A layer of PST that focuses on hiding facility-dependent details of common tasks such a job submission or data archiving.

# A. `archive` man page

## NAME

**archive** - perform basic file-handling operations on the local archival storage system.

## SYNOPSIS

**archive put** [putopts] *file1* [*file2 ...*]
**archive get** [getopts] *file1* [*file2 ...*]
**archive mv*** [mvopts] *file1* [*file2 ...*] *target*
**archive rm*** [rmopts] *file1* [*file2 ...*]
**archive ls** [lsopts] [*file1 ...*]
**archive stat** [statopts] [*name*]
**archive mkdir** [mkdiropts] *dir1* [*dir2 ...*]
**archive rmdir*** [rmdiropts] *dir1* [*dir2 ...*]
**archive chmod*** [chmodopts] *mode file1* [*file2 ...*]
**archive chgrp*** [chgrpopts] group *file1* [*file2 ...*]
**archive chown*** [chownopts] owner[:group] [*file1 ...*]
**archive migrate*** -(M|u|q) [migrateopts] *file1 file2 ...*]

## DESCRIPTION

**archive** provides users with a uniform, site-independent command interface to the local archival storage system, eliminating the need to learn the myriad syntactical variations that appear at different institutions to perform essentially identical archival file management tasks.

## COMMANDS

Each instance of **archive** performs the file-management command identified by **archive**'s first argument: **put**, **get**, **mv***, **rm***, **ls**, **stat**, **mkdir**, **rmdir***, **chmod***, **chgrp***, or **chown***. Each command accepts a set of arguments as well as a set of command-line options described in the **OPTIONS** section below. If multiple archival storage facilities are available at a particular site, all of the commands act on the the default system only, unless another system is specified with the **-asf** *host* option. The detailed behavior of the different commands is as follows:

* The current version of **archive** does not support the commands indicated with asterisks (*). Later versions will incorporate these additional commands.

**archive put** [**putopts**] ***file1*** [***file2 ...***]

Copy files *file1* [*file2 ...*] from the current working directory to the archival storage system. Options *putopts* include **-retry N**, **-asf** *host*, **-C** *path*, **-s**, **-f**, **-i**, **-p**, **-M**, **-D**, **-n**, **-t** *tarfile*, **-S**, and **-z**. *file1* [*file2 ...*] may contain wildcards * and ?.

The destination directory on the archival storage system may be specified using the **-C** *path* option. If **-C** *path* is not present, the user's archival home directory will be used.

Multiple files are combined with tar(1) before copying to the archival storage system if the **-t** *tarfile* option is specified. Files are tarred by first executing

tar cf - run10 run10.001, ..., run10.025 ¿ *tarfile*

then *tarfile* is transferred to the archival storage system.

**archive get** [**getopts**] ***file1*** [***file2 ...***]

Copy files *file1* [*file2 ...*] in the archival storage system to the current working directory. Options *getopts* include **-retry N**, **-asf** *host*, **-C** *path*, **-s**, **-f**, **-i**, **-p**, **-M**, **-n**, **-S**, **-x**, and **-z**. If wildcards * and ? are used in specifying *file1* [*file2 ...*], then double quotes must be used:

archive get "*file1* [*file2 ...*]"

A source directory on the archival storage system may be specified using the **-C** *path* option. If **-C** *path* is not specified, the user's archival home directory will be used.

Archival storage files with a .tar, .tar.gz, or .tgz extension will be extracted with the **gzip** and **tar** utilities if the **-x** option is present. For example, if *filea* is named run10.tgz, then **archive get -x** will copy run10.tgz from the archival storage system, then execute

gunzip **-f** ¡ run10.tgz | tar xf -

Specifying **-x** has no effect on files with extensions different from .tar, .tar.gz, or .tgz.

**archive mv*** [**mvopts**] ***file1*** [***file2 ...***] ***target***

Move *file1* [*file2 ...*] to the target location *target*; *file1* [*file2 ...*] and *target* are all located on the archival storage system. Options *mvopts* include **-retry N**, **-asf** *host*, **-C** *path*, **-s**, **-f**, **-i**, **-M**, and **-n**. If *target* is an existing directory, then *file1* [*file2 ...*] are moved into *target*. If *target* does not exist, **archive mv** will attempt to rename *file1* as *target*. If in this case [*file2 ...*] is present or the names *file1* and *target* are identical, **archive mv** will fail and write an error message to standard error. If wildcards * and ? are used to specify *file1* [*file2 ...*] and *target*, then double quotes must be used:

archive mv "*file1* [*file2 ...*]" "*target*"

Note: do not include both *file1* [*file2 ...*] and *target* in the same set of quotes. If wildcards are used to specify *file1* [*file2 ...*] or *target*, but not both, then quotes need only delineate the argument containing wildcards.

**archive rm*** [**rmopts**] ***file1*** [***file2 ...***]

Remove *file1* [*file2 ...*] on the archival storage system. Options *rmopts* include **-retry N**, **-asf** *host*, **-C** *path*, **-s**, **-f**, **-i**, **-r**, and **-n**. If wildcards * and ? are used to specify *file1* [*file2 ...*], then double quotes must be used:

archive rm "*file1* [*file2 ...*]"

When **-r** is specified, directories are emptied of files and removed. The user is prompted for removal of any write-protected files, except when the **-f** option is used.

**archive ls** [**lsopts**] [***file1 ...***]

List [*file1 ...*] on the archival storage system. Options *lsopts* include **-retry N**, **-asf** *host*, and other options identified by the manual page for **ls** on the archival storage system. (See **man archive_ls(3)** for a full list of *lsopts*.) If wildcards * and ? are used to specify [*file1 ...*], then double quotes must be used:

archive ls "[*file1 ...*]"

If [*file1 ...*] is not specified, then the contents of the user's home directory on the archival storage system are listed.

**archive stat** [**statopts**]

Check and report status of archival storage system. Options *statopts* include **-retry N**, **-asf** *host*, and **-s**. An exit status of 0 is returned when the system is operational; a non-zero exit status is returned otherwise. A message of the form

hh:mm:ss mm/dd/yyyy *name status* [*note*]

is printed to standard out, where hh:mm:ss mm/dd/yyyy is the current date, *name* is the name of the archival storage system, *status* is either "on-line" or "unavailable", and *note* is an optional text message that may or may not be returned.

**archive mkdir** [**mkdiropts**] ***dir1*** [***dir2 ...***]

Create directories *dir1* [*dir2 ...*] on the archival storage system. Options *mkdiropts* include **-retry N**, **-asf** *host*, **-C** *path*, **-s**, **-p**, **-m** *mode*, and **-n**.

**archive rmdir*** [**rmdiropts**] ***dir1*** [***dir2 ...***]

Remove empty directories *dir1* [*dir2 ...*] from the archival storage system. Options *rmdiropts* include **-retry N**, **-asf** *host*, **-C** *path*, **-s**, **-p**, and **-n**. If *dir1* [*dir2 ...*] are not empty, **archive rmdir** will fail and print an error message to standard error. If wildcards * and ? are used to specify *dir1* [*dir2 ...*], then double quotes must be used:

archive rmdir "*dir1* [*dir2 ...*]"

**archive chmod*** [**chmodopts**] ***mode file1*** [***file2 ...***]

Change permissions for *file1* [*file2 ...*] according to *mode*. Options *chmodopts* include **-retry N**, **-asf** *host*, **-C** *path*, **-s**, **-R**, and **-n**. *mode* may be either symbolic or absolute. See discussion of *mode* syntax in the **OPTIONS** section under **-m** *mode*.

**archive chgrp*** [**chgrpopts**] **group** ***file1*** [***file2 ...***]

Change the group ID of *file1* [*file2 ...*] to *group*, which may be a decimal group ID or a group name found in the group file. Options *chgrpopts* include **-retry N**, **-asf** *host*, **-C** *path*, **-s**, **-f**, **-h**, **-R**, and **-n**. Only an appropriately authorized user may change the group of a file that is owned by another user. Unless users are appropriately authorized, they must be a member of the specified group to change the group of a file. Unless the user is appropriately authorized, **archive chgrp** clears the set-user-ID and set-group-ID file mode bits.

**archive chown*** [**chownopts**] **owner**[**:group**] ***file1*** [***file2 ...***]

Change the owner of *file1* [*file2 ...*] to *owner*; The optional *group* argument changes the group. Options *chownopts* include **-retry N**, **-asf** *host*, **-C** *path*, **-s**, **-f**, **-h**, **-R**, and **-n**. Only an appropriately authorized user may change the owner of a file. Unless the user is appropriately authorized, **archive chown** clears the set-user-ID and set-group-ID file mode bits.

**archive migrate*** **-(M|u|q)** [**migrateopts**] ***file1*** [***file2 ...***]

Control file migration of *file1* [*file2 ...*]; Options *migrateopts* include **-retry N**, **-asf** *host*, **-C** *path*, **-s**, and **-n**. archive migrate must receive one and only one of the options -M, -u, and -q. -M migrates a

remote file or set of files. -u unmigrates a remote file or set of files, and -q prints a message indicating whether or not the file(s) are migrated.

## OPTIONS

By default, **archive** assumes all arguments beginning with a dash (-) are command-line options. Therefore, when **archive** encounters an input item of the form -?, where ? does not match any of **archive**'s valid options, **archive** will print a message to standard error and return a non-zero exit status. Because this default behavior does not allow the user to specify file or directory names that begin with a dash, you can override this default behavior by including a dash-dash (–) in the argument list. When **archive** encounters a dash-dash, it stops parsing command options and assumes all following input items are command arguments.

* archive currently does not support the options identified below with asterisks (*). There are two acceptions: **-f** is assumed, but sufficient error checking to enforce **-f** for all possible errors is not currently enabled. Also, **-m** *mode* currently only handles asolute (as opposed to sybmolic) mode specification. Later versions of archive will include all of the options and behavior listed below.

**-C** ***path***

Manage files in the archival storage directory *path*. *path* is the user's archival storage home directory by default.

**-M**

Once a file has been written or accessed on the archival storage system, migrate the file to offline media. Some centers do not support this operation. This option may prove useful for users with numerous large data files to move, which, if moved all at once, would threaten to fill the first tier of the archival storage system. By specifying **-M** the user immediately makes room for subsequent archival data transfers.

**-R***

Execute command recursively in all directories and subdirectories in the argument list. Same as **-r**. **archive chmod** only takes **-R**, since **-r** can have a different meaning for this command.

**-asf** ***host***

Archival Storage Facility: execute command on the archival storage facility *host*. This option overrides the default specified by the $ARCHIVE_HOST environment variable.

**-D**

Delete the local copy of source files involved in data transfer. If not set, all local source files are retained. Note, "source files" refers to files before they have been tarred with the **-t** option or local files that are not tarred before transfer. "Source files" are not the intermediate tarfiles generated when **-t** is envoked. Intermediate tarfiles generated with **-t** are deleted by default upon successful transfer. To override this default behavior and retain intermediate tarfiles, use **-S**.

**-S**

Save local copies of transferred tarfiles.

**-f***

Force creation, deletion, or modification of files/directories, possibly overriding permissions set on existing files/directories owned by the user. When **-f** is not present, the user is prompted interactively to override permission settings on files owned by the user. Regardless the presence of **-f**, **archive** will not attempt to change permissions on files or directories not owned by the user. When used in conjunction with **archive chgrp** or **archive chown**, the **-f** option is identical to **-s**.

**-h***

If the file is a symbolic link, change attributes of the symbolic link. If **-h** is not used, attributes of the file referenced by the symbolic link are changed.

**-i***

Force interactive confirmation by the user before overwriting or removing an existing file or directory, both on the archival storage system and local disks. If **-i** is not set and permission settings permit, **archive** will overwrite or remove existing files/directories.

**-m** ***mode****

Specify the permissions for the directory or file according to *mode*. The syntax for *mode* described here is also used by the **archive chmod** command.

File *mode*s may be either symbolic or absolute. Absolute changes to modes are specified with octal numbers, with *mode* of the form *nnn*. Here *n* is a number from 0 to 7.

An absolute mode is given as an octal number constructed from the "OR" of the following modes:

**Code**

Description

**400**

Sets read permission for owner.

**200**

Sets write permission for owner.

**100**

Sets execute (search in directory) permission for owner.

**070**

Sets read, write, and execute (search in directory) permission for group.

**007**

Sets read, write, and execute (search in directory) permission for others.

Symbolic changes to modes are specified with mnemonic characters, with *mode* of the form

[*who*] *operator* [*permissions*]

Note: if *who* is specified, there should be no space between *who* and *operator*; similarly, if *permissions* is specified, there should be no space between *operator* and *permissions*. Here *who* is one or more characters that corresponds to user, group, other, or all (**u**, **g**, **o**, or **a**); *operator* is +, -, or =, signifying the addition, removal, or absolute assignment of permissions; and *permissions* is one or

more characters that correspond to the type of permission, e.g., read (**r**), write (**w**), execution (**x**) for a given file. For example, to add read and write permission for group, *mode* should be g+rw.

Multiple symbolic modes separated by commas can be specified, although no spaces may intervene between these modes. Operations are performed in the order given. Multiple symbolic letters following a single operator cause the corresponding operations to be performed simultaneously.

By default *mode* is 777 (a+rwx) minus the bits set in the umask on the archival storage system.

**-n***

Execute **archive** in background mode and return zero exit status. If **-i** is set, **-n** is ignored.

**-p**

Create all non-existent parent directories in a specified archival storage system path. By default, if the parent directory of a target pathname does not exist, **archive** will fail and write an output error message to standard error.

**-q**

Query the migration status of a remote file. Used with **archive migrate**.

**-r**

Execute command recursively in all directories and subdirectories in the argument list. Same as **-R**. **archive chmod** only takes **-R**, since **-r** has a different meaning for this command.

**-retry N**

If **archive** fails, retry **N** times, with increasing wait time between repeated attempts. Currently **archive** sleeps for 5 seconds plus two seconds for every iteration attempt. Default is **-retry 1000**.

**-s**

Suppress printing of any output messages to standard error or standard out, except for usage errors.

**-t *tarfile***

This option handles archival storage files created with tar(1). See **DESCRIPTION** section for **archive put** above for discussion of the **-t** option.

**-u**

Unmigrate a remote file. Used with **archive migrate**.

**-x**

Extract (possibly gzipped) tarfiles after transfer from archival storage with **archive_get**.

**-z**

Use gzip(1) to compress/uncompress files transfered to/from archival storage. The compressed file will have a .gz extension. If **-t** is also set, then the output from tar(1) will be piped to gzip(1), and the resulting compressed tar file will have a .tgz extension. See **EXAMPLES** section below.

## DIAGNOSTICS

## EXAMPLES

To create directory dir1/dir2 on the data archive when dir1 doesn't exist:

```
archive mkdir -p dir1/dir2
```

To get a listing of all files that end with .tex in directory d1 on the data archive:

```
archive ls "d1/*.tex"
```

To put local files f1 and f2 into a zipped tarfile files.tgz into directory dir1 on the mass store and delete the local copies:

```
archive put -C dir1 -D -z -t files.tar f1 f2
```

To tar a local directory localdir into and put the tar file (tarfile.tar) in remote directory remotedir:

```
archive put -C remotedir -t tarfile.tar localdir
```

To get and extract files from tarfile files.tar in mass-store directory dir1, and save (not delete) the tarfile in the local directory:

```
archive get -C dir1 -x -S files.tar
```

To get a file from directory dir1 and make sure that the remote file is still/again migrated to slow storage:

```
archive get -C dir1 -M file
```

## ENVIRONMENT

The following environment variables are checked by **archive**: $ARCHIVE_HOST, $ARCHIVE_HOME, and $ARCHIVE_PROBE. If set, **archive** overrides default settings established during installation.

Each environment variable is a comma-separated list (with no white space), representing multiple archival storage facilities. Currenlty all **archive** commands operate on only one system and use the first element of each list, which is interpretted by **archive** as the default. A different system may be specified with the **-asf** option. In the future, **archive get**, **stat**, and **ls** will be modified to operate on multiple systems, sequencing through the entries in the lists for these environment variables.

**$ARCHIVE_HOST**

List of archival storage systems available at this site. **-asf** *host* overrides this environment variable.

**$ARCHIVE_HOME**

List of archival storage system home directories associated with $ARCHIVE_HOST. $ARCHIVE_HOME must contain either only one element, which is used for all elements of $ARCHIVE_HOME, or a separate element for each system specified by $ARCHIVE_HOST.

**$ARCHIVE_PROBE**

List of flags ("yes" or "no") that indicate whether or not **archive** will attempt to check the existence, type, and size of files on the archival storage system before and after transfer so that error checking is possible. A reason for setting $ARCHIVE_PROBE="no" could be failure in accurate or complete reporting of file information by the archival storage system.

## FILES

## NOTES

**archive** is a command-line translation tool in the TUSC layer of the Practical Supercomputing Toolkit (PST). By using **archive** (and other elements of PST), users develop scripts and techniques that work, without modification, at all institutions supported by PST.

Centers currently supported include ERDC, NAVO, ASC, AHPCRC.

## SEE ALSO

archive_put(3),
archive_get(3),
archive_mv(3)*,
archive_rm(3)*,
archive_ls(3),
archive_stat(3),
archive_mkdir(3),
archive_rmdir(3)*,
archive_chmod(3)*,
archive_chgrp(3)*,
archive_chown(3)*.

## AUTHOR

Written and Copyright (C) 2001 by Joseph Werne, Michael Gourlay, and Chris Bizon. See www.pstoolkit.org.

## HISTORY

Documentation v1.0 assembled Friday, July 13, 2001. Revised (v1.1) Sep 18, 2001.