To: Distribution

From: Lindsey Spratt

Date: 05/04/78

Subject: Copying and moving directories

Introduction

This MTB describes two related commands, move_dir and copy_dir. These commands move or copy a directory and its subtree. There follows a discussion of some of the issues, and the draft command descriptions.

Issues

These commands are proposed as existing separately from the move and copy commands because there is little overlap in the functions of those commands. For instance, move_dir and copy_dir must recurse through the hierarchy and they must deal with links as opposed to link targets, both are activities foreign to copy and move. Also, creating and deleting. entries are operations which have one set of commands for directories and another for non-directories. It seems appropriate to follow this precedent .

Consistency with the behavior of the move and copy commands is seen as an important criterion in the designs of move_dir and copy_dir, particularly in the choice of defaults. One difference in defaults concerns links. The directory commands do not chase links, unless asked. Another difference concerns the handling of multiple names. Copy_dir copies all names by default, while copy takes only the primary name.

The -force control argument could force a number of things:

1. Continuation of execution on the existence of target_dir.
2. Deletion of old segments (in target_dir) in certain name duplication situations.
3. Deletion of entries in source_dir regardless of the safety switch.

It is only proposed to do the first of the above items, for fear

of making the control argument too strong.

When copying of a particular entry fails, the chosen action is to ignore that entry (while issuing a warning) and continue with the next one.

Command descriptions

See following pages.

Please send comments to:

Spratt.Multics,

or

Lindsey Spratt
Honeywell Information Systems
575 Tech Square
Cambridge, Mass. 02139

Or call:

(617) 492-9321
HVN 261-9321

Name: copy_dir,cpd

The copy_dir command copies a directory and its subtree to another point in the hierarchy.

Usage

copy_dir source_dir target_dir {-control_args} {-entry_type_keys}

where:

1. source_dir
   is the pathname of the directory to be copied.

2. target_dir
   is the pathname of the copy of the source_dir. If target_dir is not specified, the copy is placed in the working directory with the entryname of source_dir. If the target_dir does not exist it is created.

3. control_args
   can be chosen from the following list of control arguments:

   -replace, -rp
   the contents of target_dir existing before the copying begins are deleted. If target_dir is non-existent or empty, this control argument has no effect. The default is to append the contents of the source directory to the target directory if it already exists.

   -no_link_translation, -nlt
   copies links with no change. The default is to translate links being copied. Translation changes the link pathname to the new copy of the target of the old link. Translation need only occur when a link and its target are both in the source_dir.

   -acl
   gives the ACL on the source_dir entry to its copy in target_dir. Although initial ACL's are still copied, they are not used in setting the ACL of the new entries.

-force
continues execution when target_dir already exists without asking the user.

-primary, -pri
copies only primary names.

-brief, -bf
suppresses the warning messages "Bit count inconsistent with current length ..." and "Current length is not the same as records used ...".

-chase
copies the target of a link. The default is not to chase links. Chasing the links eliminates link translation.

4. entry_type_keys
The entry_type_key controls what is copied. Including a key in the mode specification directs copy_dir to copy all instances of that entry type. If no entry_type_key is given, all entries are copied. If any entry_type_key is given, only those entry types specified are copied. The keys are:

-branch
-directory
-file
-link
-msf
-non_null_link
-segment

If one or more entry_type_keys are specified, but not the -directory key, the subtree of source_dir will not be followed.

Access requirements

Status permission is required for source_dir and all of the directories in its tree. Status permission is required for the directory containing source_dir. Read access is required on all files under source_dir. Append and modify permission is required for the directory containing target_dir if target_dir doesn't exist prior to the invocation of the copy_dir command. Modify and append permission is required on target_dir if it already exists. This command does not force access.

Access provision

If the -acl control argument is not specified, the system default ACL's are added, then the initial ACL for the containing directory is applied (which may change the system supplied ACL). Initial ACL's are always copied for the current ring of execution, although they aren't used by the copy_dir command if -acl is given.

Existence of target dir

If target_dir already exists, the user is so informed and asked if processing should continue. If target_dir is contained in or contains source_dir, an appropriate error message is printed and control is returned to command level. Otherwise, the contents of source_dir are either appended to or replace the contents of target_dir. (See the -replace control argument.)

Star and equals conventions

The star and equals conventions can be used. The star convention will match only directory names and copy them. Matching names associated with other storage types will be ignored.

Name duplications

Since two entries in a directory cannot have the same entry name, this command takes special action if the entryname of the entry being copied already exists in the directory specified by target_dir. If the entry is a directory, it is dealt within the same fashion as duplication between source_dir and target_dir is handled, unless the existing entry in target_dir is not also a directory. In this case the entryname duplication is treated the same as non-directory entries. The procedure for non-directory entries is the standard system technique.

If the -replace control argument is specified or target_dir does not exist, name_duplication will not occur.

Link translation

If part of the tree is not copied, problems with link translation may occur. If the link in the source_dir tree was in the part of the tree not copied, there may be no corresponding entry in the target_dir tree. Hence, translation of the link (presumably originally non-null) will cause the link to become null.

Name: move_dir,mvd

The move_dir command moves a directory and its subtree, including all of the associated attributes, to another point in the hierarchy. Links are translated; that is, if the target of a link being moved is also being moved, then the link pathname of the moved link points to the moved target.

Usage

move_dir source_dir target_dir {-control_args}

where:

1. source_dir
   is the pathname of the directory to be moved.

2. target_dir
   is the new pathname for source_dir. If the entryname is different from one already on source_dir, it is added to the existing names. If target_dir is not given, source_dir is moved to the working directory and given the same entryname.

3. control_args

   -brief, -bf
   suppresses the printing of warning messages.

   -force
   continues execution when target_dir already exists, without asking the user.

   -replace, -rp
   the contents of target_dir existing before the copying begins are deleted. If target_dir is non-existent or empty, this control argument has no effect. The default is to append the contents of the source directory to the target directory if it already exists.

4. entry_type_keys
   The entry_type_key controls what is moved. Including a key in the mode specification directs move_dir to copy all instances of that entry type. If no entry_type_key is given, all entries are moved. If any entry_type_key is given, only those entry types specified are moved. The keys are:

-branch,
-directory,
-file,
-link,
-msf,
-non_null_link,
-segment

If one or more entry_type_keys are specified, but not the -directory key, the subtree of source_dir will not be followed.

Access requirements

Status and modify permission is required for source_dir and all of the directories in its tree, and its containing directory. If target_dir doesn't exist, append permission is required for its containing directory. If it does exist, modify and append permission for target_dir is required. This command does not force access.

Access provision

The access control language associated with source_dir is moved to target_dir.

If target_dir already exists, the user is so informed and asked if processing should continue. If target_dir is contained in or contains source_dir, an appropriate error message is printed and control is returned to command level. Otherwise, the contents of source_dir are either appended to or replace the contents of target_dir. (See the -replace control argument.)

Star and equals conventions

The star and equals conventions can be used.

Name duplications

Since two entries in a directory cannot have the same entry name, this command takes special action if the entryname of the entry being copied already exists in the directory specified by target_dir. If the entry is a directory, it is dealt within the same fashion as duplication between source_dir and target_dir is handled, unless the existing entry in target_dir is not also a

directory. In this case the entryname duplication is treated the same as non-directory entries. The procedure for non-directory entries is the standard system technique.

If the -replace control argument is specified or target_dir does not exist, name_duplication will not occur.

Link translation

If part of the tree is not copied, problems with link translation may occur. If the link in the source_dir tree was in the part of the tree not copied, there may be no corresponding entry in the target_dir tree. Hence, translation of the link (presumably originally non-null) will cause the link to become null.