

---

# System Administration Toolkit: Migrating and moving UNIX filesystems

Skill Level: Intermediate

[Martin Brown \(mc@mcslp.com\)](mailto:mc@mcslp.com)  
Freelance Writer  
Consultant

03 Jul 2006

Learn how to transfer an entire file system on a live system, including how to create, copy, and re-enable the new file system. If you have a UNIX® disk or system failure or simply fill up your file system, then you need to create a new partition and file system and copy over the contents. You might even need to mount the new partition in place to preserve the location of vital files and components. To add further complications, you need to do this on a live system, where you'd need to preserve file permissions, ownership, and possibly named pipes and other components. Effectively transferring these components and retaining all of this information is a vital part of the migration process.

## About this series

The typical UNIX® administrator has a key range of utilities, tricks, and systems he or she uses regularly to aid in the process of administration. There are key utilities, command-line chains, and scripts that are used to simplify different processes. Some of these tools come with the operating system, but a majority of the tricks come through years of experience and a desire to ease the system administrator's life. The focus of this series is on getting the most from the available tools across a range of different UNIX environments, including methods of simplifying administration in a heterogeneous environment.

## Moving a UNIX directory or filesystem

There are many occasions when you might need to move a UNIX filesystem from

one device or hard disk partition to another, or when you need to move content from a file system to free up space, creating a new file system in the process. There might be an impending fault with the device, or you might simply be running out of space. You can do this on a system that is running in single-user mode, or on a *live* and running system where you need to ensure that the files are available during the move.

In the case of the latter situation -- migrating a live file system -- it could be that you need to move an application while the application is running. In either case, you might have to move a system-related file system (such as /usr or /var), and whether you are running in single-user mode or not, the file system might actually be in active use.

In any file system movement operation, you must ensure that the data has been correctly copied to the new destination. As a typical example, imagine that you have a system with a file/disk layout and usage like that in [Listing 1](#).

### Listing 1. Example file/disk layout and usage

Filesystem	1K-blocks	Used	Available	Use%	Mounted on
/dev/sda3	7692908	6467756	1225152	84%	/
udev	517560	184	517376	1%	/dev
/dev/hda1	115377640	1743668	107773060	2%	/var/lib/mysql
/dev/hdb1	115380192	14604460	94914696	14%	/export/data
/dev/hde1	96132940	3962940	87286644	5%	/export/home
/dev/hde4	22960280	133484	21660480	1%	/tmp
none	517560	0	517560	0%	/dev/shm

As you can see in [Listing 1](#), the root file system is 84 percent full and you should probably move a directory from the root file system that is not already on a different partition to a new partition, or device, to ensure that the root file system does not fill up.

It is best to move a single directory with a specific purpose that has the most significant impact on the file system that you want to free up space on. For example, in this case, you might want to put the /var or /usr directory onto its own file system. Moving multiple, smaller, directories is not as efficient. You want a single and easy to move directory to free up space.

In some situations, you will be moving or copying an existing partition to a new location (for example, during a potential device failure). Here, the choice of the directory or existing file system already has been made for you.

Throughout this article, I assume that you have already added a new hard disk device, or have a spare disk device or partition to use as the destination for the directory or file system.

## Quick guide to moving data

The basic sequence for moving a file system, or for placing an existing directory onto a new file system, is as follows:

- Choose the data to be copied.
- Create a new partition.
- Create a new file system on the partition.
- Mount the partition on a temporary directory.
- Copy the data to the temporary directory.
- Rename the original directory.
- Create the new mount point.
- Mount the file system.

Ideally, at the end of this, you should reboot, although this isn't always necessary or possible; you should certainly try to reboot at the earliest opportunity.

Let's take a closer look at each step of the sequence.

## Creating a new partition and file system

Before you create a new partition and file system, first ensure that the available size and space matches the size of the files/directory that you are going to be placing on the new file system. The easiest way to do this is to use the `du` tool to determine the current file/directory size. For example, if you are going to move the `/usr` directory from the example system, you would use the command shown in [Listing 2](#).

### Listing 2. Moving the `/usr` directory from the example system

```
$ du -sk /usr
3923068 /usr
```

The `-s` option ensures that it generates a summary of the whole directory, rather than the contents, and the `-k` option ensures that you get the directory size in kilobytes. From the above, you can determine that you need a new partition that is at least 4GB in size.

If you are moving an existing file system to a larger partition, (rather than a single directory within an existing file system) then use the output from `df` to determine the current file system size.

Once you know the size, you should create a partition that is ideally 25 percent larger than the size you need, and certainly no less than 10 percent larger. For the example file system above, you need a new partition that is at least 5GB in size.

To create a new partition and file system, you need to use the tools that are appropriate for your operating system. For example:

- Under Linux®, you need to use `fdisk` to configure the partitions on a physical hard drive and one of the `mke2fs` or `mkfs.*` creation commands to create a suitable file system on the new partition.
- On a Solaris SPARC box, use `format` to create or configure the partition. On a Solaris x86 box, you might also need to use `fdisk` to configure the disk partition before using `format` to configure the Sun partition table. You can then use `newfs` (or one of the file system specific commands) to create the file system.
- Under AIX®, use `mkvg`, `chpv`, or some other volume management tool to create a suitable partition or volume group for your new partition. You can then use `crfs` to create the file system.

With the file system in place, create a new directory under which you can mount the new file system while the data is copied. For example, I prefer to create a new file system within a similar location and with a name that identifies the file system as temporary.

With the `/usr` example above, I would create a new directory, `/mnt/usr.tmp`, where the new file system would be mounted.

## Copying the information

Actually copying the information is probably the simplest part of the process, but choosing the right tool can make a big difference to how effectively and efficiently the information is transferred. You should, however, take care to ensure that you are copying files to the correct location; you want to avoid overwriting existing data.

There are many different methods available, but the two primary solutions are to use the basic `cp` tool or the `tar` tool. The former is a very quick method but, on some operating systems, you might experience problems with non-standard files, such as

pipes and devices. The tar tool can be a slower alternative, but it is more reliable when transferring other file types and provides more visual feedback about the copy process.

Whichever method you use, you should ensure that any files on the source file system are not actively being updated. Remember that any copy you make only contains the data that was in the individual files at the point in time the copy was made. If files are being updated on the file system at the time, the files on the new file system might be incomplete and corrupt.

## Using cp

The cp tool supports a recursive copying option (`-r`) that copies all the files, directories, and the files within those directories to the destination. You can also use the `-p` option to preserve ownership and permissions on the copied files. If you want to ensure the new file system retains the security details of the source, this is vital. There are few situations when this is not a requirement.

To copy the files from an existing directory to the new location, follow these steps:

1. Change to the new target directory and confirm that you are in the right place (see [Listing 3](#)).

### Listing 3. Changing the target directory

```
$ cd /mnt/usr.tmp
$ pwd
/mnt/usr.tmp
```

2. Copy the files from the old directory to the current (new) directory, as shown in [Listing 4](#).

### Listing 4. Copying the files to the new directory

```
$ cp -pr /usr/* .
```

You should perform a quick verification that the files have copied over. A simple `ls` will give an indication (see [Listing 5](#)). You'll examine more detailed checking methods later.

### Listing 5. Verifying that the files have copied over

```

$ ls -l /usr
total 238
drwxr-xr-x  2 root    bin          1024 Apr 20 13:11 4lib/
lrwxrwxrwx  1 root    root          5 Apr 20 12:40 5bin -> ./bin/
lrwxrwxrwx  1 root    root          9 Apr 20 12:34 X -> ./openwin/
drwxr-xr-x  6 root    bin           512 Apr 20 12:42 X11/
lrwxrwxrwx  1 root    root          3 Apr 20 12:41 X11R6 -> X11/
lrwxrwxrwx  1 root    root          10 Apr 20 13:03 adm -> ../var/adm/
drwxr-xr-x 10 root    bin           512 Apr 20 12:59 apache/
drwxr-xr-x  8 root    bin           512 Apr 20 12:47 apache2/
drwxr-xr-x  8 root    bin           512 Apr 20 12:53 appserver/
drwx----- 8 root    bin           512 Apr 20 12:53 aset/
drwxr-xr-x  4 root    bin        16384 Apr 20 13:17 bin/
drwxr-xr-x  4 root    bin           512 Apr 20 12:33 ccs/
...
lrwxrwxrwx  1 root    root          10 Apr 20 12:32 tmp -> ../var/tmp/
drwxr-xr-x  4 root    bin        2048 Apr 20 13:00 ucb/
drwxr-xr-x  4 root    bin           512 Apr 20 13:17 ucbinclude/
drwxr-xr-x  3 root    bin        1024 Apr 20 13:17 ucblib/
drwxr-xr-x  7 root    bin           512 Apr 20 13:03 vmsys/
drwxr-xr-x  5 root    bin           512 Apr 20 12:44 xpg4/
drwxr-xr-x  3 root    bin           512 Apr 20 12:40 xpg6

```

If you discover that some files have not been copied over correctly or if symbolic links and other special filetypes are not copying properly, then you might want to try tar.

## Using tar

The tar tool offers a number of benefits over cp. First and foremost, it tends to be more reliable for non-standard file types. Second, because it can provide visual feedback on the files being copied, it can be a more comforting method of copying the files across to the new system. Last, but certainly not least, you can use tar to create an archive file of the file system, which can act as a backup of your source directory/file system in event of a problem. On a user file system with a computer in single-user mode, the tar file method can be used to create the new file system content without mounting the new destination at a temporary directory mount point.

The best way to use tar for copying is to pipe the file created by tar through another tar in the new location that is extracting the files. An example is shown in [Listing 6](#).

### Listing 6. Using tar for copying

```

$ cd /usr
$ tar cfp - ./* |(cd /mnt/usr.tmp; tar xvpf -)

```

The `c` argument in the first tar tells tar to create an archive. The `v`, `f`, and `p` options specify verbose output (prints the files being added/extracted), writes/reads from a

file (instead of a tape device), and retains permissions and ownership, respectively.

The line in [Listing 6](#) works, because the second half of the pipe first changes to the target directory before reading the .tar file that is being created in the first half of the pipe from standard input.

If you want to create a .tar file and use this, rather than performing the direct copy, you must have a file system capable of holding all the files from your source directory. [Listing 7](#) shows the process for copying.

### Listing 7. Process for copying

```
$ cd /usr
$ tar cfp /tmp/usr.tar ./*
$ cd /mnt/usr.tmp
$ tar xvfp /tmp/usr.tar
```

Whichever solution you use, you should get a report as each file is copied into, or out of, the archive, providing you use the `v` command-line option (see [Listing 8](#)).

### Listing 8. Printout of copy process

```
a ./4lib/ 0K
a ./4lib/libX.so.1.0 symbolic link to ./libX11.so.4.3
a ./4lib/libX11.so.4.3 216K
a ./4lib/libXaw.so.4.0 208K
a ./4lib/libXmu.so.4.0 72K
a ./4lib/libXol.so.3.1 1056K
a ./4lib/libXt.so.4.1 264K
a ./4lib/libce.so.0.0 48K
a ./4lib/libdeskset.so.0.1 64K
a ./4lib/libdga.so.1.0 40K
a ./4lib/libhelp.so.1.0 24K
a ./4lib/libolgx.so.3.1 56K
a ./4lib/libtt.so.1.1 848K
a ./4lib/libttstub.so.1.1 32K
a ./4lib/libxview.so.3.73 1328K
a ./4lib/libdl.so.1.0 symbolic link to ../../lib/libdl.so.1
a ./4lib/libc.so.1.9 403K
a ./4lib/libc.so.2.9 402K
...
```

## Using tar and direct file system exchange

To copy the contents of a file system to a new location without creating a temporary mount point, you must first have the space to hold all of the files from your source directory.

If replacing a directory with a new file system:

1. Create the new partition and file system as before.
2. Create a .tar file of the source directory (see [Listing 9](#)).

### **Listing 9. Creating a .tar file**

```
$ cd /home
$ tar cfvp /tmp/home.tar
```

3. Rename the source directory (see [Listing 10](#)).

### **Listing 10. Renaming the source directory**

```
$ cd ..
$ mv home home.old
```

4. Create the directory and set the same permissions and ownership as the original directory.
5. Mount the new file system on the new directory.
6. Extract the .tar file (see [Listing 11](#)).

### **Listing 11. Extracting the .tar file**

```
$ cd home
$ tar xvfp /tmp/home.tar
```

If you are changing the partition for an existing file system (shown in [Listing 10](#)), then the sequence is similar, but you unmount the existing file system rather than renaming the directory (see [Listing 12](#)).

### **Listing 12. Changing the partition for an existing file system**

```
$ cd ..
$ umount /home
```

This option is still safe, because you have a complete copy of the source file system on the old partition. You haven't deleted the contents or source material in either solution.

## Verifying the copy

Whether you use `cp` or `tar`, you should always verify the copy has completed successfully. Although `tar` confirms the files that are copied (with the `v` option), you should ensure that the files are correctly recreated on the new file system.

First, check the output of `du` on both the old and the new system (see [Listing 13](#)).

### Listing 13. Checking the output of `du`

```
$ du -sk /usr
3923068 /usr
$ du -sk /mnt/usr.tmp
3923068 /mnt/usr.tmp
```

The two numbers should be identical; however, depending on the type of the new file system and file and directory allocation sizes on the old file system/directory and the new one, you might notice a slight discrepancy with the size.

Another good test is just to compare the number of files/directories on the source and destination. You can do this with a simple `find` command, as shown in [Listing 14](#).

### Listing 14. Comparing the number of files/directories on source and destination

```
$ find /usr |wc -l
347001
$ find /mnt/usr.tmp |wc -l
347001
```

Another good test, if you are copying file system to file system, is to compare inode numbers; these show how many inodes have been allocated. If you are copying from one file system to another of exactly the same type, the number of inodes used should be identical. Use `df`, with the `-i` command-line option, to get the inode statistics or, on traditional UNIX systems, the bare `df` output shows the number of 'files' created.

The example in [Listing 15](#) is from a Linux system.

### Listing 15. Comparing inode numbers

```
$ df -i /usr
```

```
Filesystem          Inodes   IUsed   IFree  IUse%  Mounted on
/dev/sda3           977280  411959  565321   43%   /
```

The example in [Listing 16](#) is on Solaris.

### Listing 16. Comparing inode numbers on Solaris

```
$ /bin/df /usr
/usr                (/dev/dsk/c0t0d0s3 ): 9076010 blocks   863695 files
```

## Updating your system to reflect the new organization

By this stage, you should have a new file system that contains a copy of the file system or directory that you are moving to a new file system. You should now update your system files (particularly the file system mount information) to reflect the new structure. This information is stored in `/etc/fstab`, `/etc/vfstab`, or might be available through a specific administration tool, such as SAM on HP-UX.

If you have been migrating a directory within an existing file system:

1. Rename the original directory.
2. Create a new directory.
3. Use `chown` and `chmod` to set the ownership and permissions on the new directory.

Ideally, you should now reboot your system to ensure that that the new layout is used. You must reboot if you are migrating a file system from one device to another. It is unlikely you will be able to unmount the existing file system, especially if it is a system directory (in other words, one under `/var` or `/usr`).

If a reboot is not possible, manually mount the new system in the new mountpoint and then reboot as soon as possible.

Once you have rebooted and confirmed everything is working properly, you can delete the old directory or reuse the old partition.

## Summary

There are many situations where copying a UNIX filesystem, live or otherwise, is

required. It could be because you are running out of space, requiring a larger partition for your file system to enable installation of software, or even an impending hardware fault. In any of these situations, you need to use the techniques covered in this article to copy over the existing files to the file system.

Such a copy is not without its traps -- copying a live file system can be risky, especially if there are open files. You should also be careful to ensure that you don't accidentally overwrite a partition, or existing files, with the files you are trying to copy. However, as you've seen here with some careful thought, you can effectively migrate files reliably to take advantage of more space, even on a live system.

# Resources

## Learn

- [System Administration Toolkit](#): Check out other parts in this series.
- ["Using ReiserFS with Linux"](#) (developerWorks, April 2006): This article provides a guide to setting up a ReiserFS filesystem, which you could then migrate data to using the tips in this article.
- ["Make UNIX and Linux work together"](#) (developerWorks, April 2006): This article gives information on sharing file systems and other data. The techniques in this article could be used to effectively migrate data between two systems.
- [AIX® and UNIX articles](#): Check out other articles written by Martin Brown.
- Search the AIX and UNIX library by topic:
  - [System administration](#)
  - [Application development](#)
  - [Performance](#)
  - [Porting](#)
  - [Security](#)
  - [Tips](#)
  - [Tools and utilities](#)
  - [Java technology](#)
  - [Linux](#)
  - [Open source](#)
- [AIX and UNIX](#): The AIX and UNIX developerWorks zone provides a wealth of information relating to all aspects of AIX systems administration and expanding your UNIX skills.
- [New to AIX and UNIX](#): Visit the New to AIX and UNIX page to learn more about AIX and UNIX.
- [AIX 5L™ Wiki](#): A collaborative environment for technical information related to AIX.
- [Safari bookstore](#): Visit this e-reference library to find specific technical resources.

- [developerWorks technical events and webcasts](#): Stay current with developerWorks technical events and webcasts.
- [Podcasts](#): Tune in and catch up with IBM technical experts.

### Get products and technologies

- [IBM trial software](#): Build your next development project with software for download directly from developerWorks.

### Discuss

- Participate in the [developerWorks blogs](#) and get involved in the developerWorks community.
- Participate in the AIX and UNIX forums:
  - [AIX 5L -- technical forum](#)
  - [AIX for Developers Forum](#)
  - [Cluster Systems Management](#)
  - [IBM Support Assistant](#)
  - [Performance Tools -- technical](#)
  - [Virtualization -- technical](#)
  - [More AIX and UNIX forums](#)

## About the author

Martin Brown

Martin Brown has been a professional writer for more than seven years. He is the author of numerous books and articles across a range of topics. His expertise spans myriad development languages and platforms -- Perl, Python, Java™, JavaScript, Basic, Pascal, Modula-2, C, C++, Rebol, Gawk, Shellsript, Windows®, Solaris, Linux, BeOS, Mac OS X and more -- as well as Web programming, systems management, and integration. He is a Subject Matter Expert (SME) for Microsoft and regular contributor to ServerWatch.com, LinuxToday.com, and IBM developerWorks. He is also a regular blogger at Computerworld, The Apple Blog, and other sites. You can contact him through [his Web site](#).

## Trademarks

IBM, AIX, and AIX 5L are registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.