Chapter 6

Configuring Disk and Swap Space

For a variety of reasons, users often accumulate files without realizing how much disk space they are using. This is not so much a problem for workstation users, but it is a real problem for servers, where multiple users must share system resources. IRIX provides a number of utilities to help you manage disk usage.

You also may need to allocate disk space for the system to use as swap space. Swap space is disk space allocated to be used as medium-term memory for the operating system kernel. Lack of swap space limits the number and size of applications that may run at once on your system, and can interfere with system performance.

This section provides you with some background information and tips on the tools and options available to you. For specifics on setting and maintaining disk quotas, see the *IRIX Admin: Disks and Filesystems* Guide.

Useful Disk Usage Commands

The du(1) Command

The *du*(1) command shows specific disk usage by file or directory anywhere on the system. The size of the files and directories can be shown in 512-byte blocks, or in 1K blocks if you use the **-k** flag. For more complete information, consult the *du*(1) reference page.

The df(1) Command

The *df*(1) command shows the free space remaining on each file system on your workstation. Free space is shown for all file systems, whether they are local or NFS-mounted. The amount of free space is displayed in 512-byte blocks, or in 1K blocks if you use the **-k** option. For more complete information, consult the *df*(1) reference page.

The quot(1M) Command

The *quot*(1M) command displays the number of kilobytes of disk usage for each user in a specified file system. You can use the output of this command to mail your users, notifying them of their disk usage. For complete information, see the *quot*(1M) reference page.

The diskusg(1) Command

Part of the system accounting package is the *diskusg*(1) command. Like *quot*, this utility reports the disk usage of each user on your system. The difference is that *diskusg* is typically used as part of the system accounting package with *dodisk* rather than as a standalone command. For complete information on using *diskusg*(1), see the *diskusg* reference page.

File Compression and Archiving

One way to minimize disk usage is to encourage your users to archive or compress their files. Compression works best for files that are rarely used but that should not be discarded. Users can achieve some disk savings by using the *compress*(1) utility. Compressing a file allows you to leave it on the system, but can reduce the size of the file by as much as 50%. Compressed files have the suffix **.Z** and should not be renamed. Compression and archiving can be used together for maximum disk space savings.

If the files in question are used infrequently, consider archiving them to tape, floppy, or some other archive media. This maintains protection if you need the files later, but regaining them is slightly more difficult than if they were compressed.

The quotas(4) Subsystem

The *quotas*(4) subsystem exists to allow you to deal with severe disk usage problems. Using this subsystem, you can place a maximum disk usage quota on each user on your system. For complete information about this subsystem, see the *quotas*(4) reference page.

In general, it is your job as the site administrator to set disk use policies, establishing and enforcing quotas if necessary. You should publish clear guidelines for disk use, and notify users who regularly exceed their quotas. It is also a good idea to impose quotas on

the use of temporary directories, such as */tmp* and on all anonymous ''guest'' accounts, such as *guest* and *uucp*. If your root file system reaches 100% capacity, your system may shut down and inconvenience your users.

You should be as flexible as possible with disk quotas. Often, legitimate work forces users to temporarily exceed disk quotas. This is not a problem as long as it is not chronic.

Do not, under any circumstances, remove user files arbitrarily and without proper warning.

A typical scenario is when all the users on the system know they should try to limit disk use in their home accounts to about 20 MB (about 40,000 512-byte blocks). User *norton* consistently uses more than twice this limit. These are the steps you take to alleviate the problem:

1. Try to meet with the user and find out why he or she is using this much disk space. There may be legitimate reasons that require you to reconsider the disk use policy and perhaps increase the amount of available disk space.
2. If the user is merely saving an inordinate number of outdated files, suggest that he or she back up the files onto tape and remove them from the system. For example, many users save electronic mail messages in enormous mailboxes for long after the messages are useful. Saving the files to tape keeps them handy, while saving disk space.
3. If you cannot meet with the person, or cannot discuss the matter with them, try sending them electronic mail.

   If you use a script that automatically checks disk use (with *diskusg*) and sends mail to users who exceed their quotas, note that people get used to these messages after some time and start to ignore them. Send the particular user a personal message, stating that you need to discuss the situation with him or her.
4. Sometimes a user is not aware that data is available elsewhere on the system, or on other accessible workstations at the site. A user may have personal copies of site-specific tools and data files. Work with the user to reduce this kind of redundancy.
5. Make sure the user is still active on the system. Sometimes people leave an organization, and the site administrators are not immediately informed.

Also, the user may not need the account on a particular workstation any longer and may not have cleaned up the files in that account. To see if this is the case, check the last time the user logged in to the system with the *finger*(1) command:

```
finger norton
```

Among other information, *finger* displays the date and time the user last logged in to the system. This information is read from */etc/wtmp*, if it exists.

6. If in an extreme case you must remove a user's files, back them up to tape before removing them. Do not take this step lightly. Removing user files unnecessarily can disrupt work and engender ill-will from your coworkers. Make sure you give the user plenty of advance notice that you are going to copy the files to tape and remove them from the system.

   As an added precaution, you may wish to make two copies of the tape and send one copy to the user whose files you remove. Make sure you verify that the tapes are readable before you remove the files from the system.

Managing Disk Space with NFS

If your system is running the optional Network File System (NFS) software, you can use this product to reduce disk consumption on your workstations by exporting commonly used directories. When a system exports a directory, it makes that directory available to all systems running the NFS software. For example, if you have 10 workstations on your network and each workstation is storing 5 MB of online reference pages and release notes, you can eliminate 45 MB of wasted disk space by designating one workstation to be the reference page server and exporting the */usr/man* and */usr/catman* directories. All other workstations can remove those files and mount them remotely from the server. Since NFS mounts take up no disk space on the client workstation, that disk space is available for other uses.

Another option is to mount free disk space from another system. This option works best when there is an uneven load across several systems. For example, if one workstation is using only 25% of its available space and other workstations are above 90%, it may be useful to mount a file system to even out the load.

Used wisely, your network can save a tremendous amount of disk space through this strategy. Be aware, though, that the drawback to this strategy is that if your server must be rebooted or serviced, no one will be able to access the files and programs that are mounted from that server while it is off-line.

Managing Disk Space with Disk Partitions

An extreme method of enforcing disk use quotas is to create a disk partition and file system for each user account and to mount the file system on an empty directory as the user's home directory. Then, when users run out of disk space, they will not be able to create or enlarge any files. They will, however, still be able to write their files to */tmp*, */usr/tmp.O*, and */var/tmp*, unless those directories are also full. When users attempt to write or create a file that takes them over their limit, they receive an error message indicating that no disk space is left on the device.

This method of disk control is not generally recommended. It requires a great deal of preparation and maintenance on the part of the Administrator, and is not easily modified once in place. Additionally, fragmenting your disk into many small partitions reduces the total amount of available disk space and produces a performance overhead on your system. In this case, the disk controller must write a user's files within only one small partition, instead of in the next convenient place on the disk.

You should consider this method of disk space control only if your system is so overloaded and your users so obstinate that all other measures have failed. If your */usr* partition is chronically 100% full, the system is halting operations daily due to lack of disk space, and there is no way to increase your disk space, then you may want to consider this method.

Reducing Wasted Disk Space

Sometimes there is a great deal of wasted disk space on a system. When you are low on space, check to make sure that large files have not accumulated in the temporary directories or in the administrative directories.

The */var/adm* directory structure is notorious for accumulating large log files and other large files that are useful for debugging problems, but which otherwise use up a tremendous amount of space. The */var/adm/crash* directory may be storing several images of the IRIX kernel (from past system failures) and copies of the entire core memory (*vmcore*). These files can take up large amounts of space (many megabytes) and should be archived on tape and removed from your system.

If you have system auditing enabled, be aware that this facility generates very large audit trail record files, and these will need to be archived to tape on a regular (perhaps daily) basis.

Also, check any UUCP or *ftp* directories you may have available, and generally look for *core* files, and any other large and unnecessary files that may exist on your system.

Swap Space

The IRIX operating system uses a portion of the disk as *swap space* for temporarily saving part or all of a user's program when there is not enough physical memory to contain all of the running programs. If you run many very large programs, you might run out of swap space. For a complete discussion of the dynamics of paging and swapping, see "Checking for Excessive Paging and Swapping" on page 199.

Use *sar -p*, *sar -w* , *sar -q*, *swap -s*, and *swap -l* to monitor paging and swap space use. If you find that you are running out of swap space, two solutions are available: you can add more memory, or you can add more swap space. Adding swap space does not improve the performance of large programs, but it permits them to run successfully.

Under *sar*(1), swapping of whole processes is reported with the *-w* flag; but this is not what causes performance problems, the problems come from the excess or slow I/O involved in paging.

IRIX allows programs occupying more space than the system limit to run, since each program is only partially loaded into memory at any given time. One of the effects of this policy is that IRIX has to preallocate swap space based on likely future usage, and sometimes this prediction is incorrect. When the swap space is actually needed, IRIX allocates the most convenient available space, not the specific space allocated. So the physical allocation is separate from the accounting allocation.

If your system preallocates all your swap space, but the space has not yet been used, it may appear that your system is running out of swap space when it is not. It is possible that your system has simply preallocated the rights to future swap space to existing processes, and no new processes can allocate space due to the strict swap space accounting in this version of IRIX.

In previous versions of IRIX, available swap space accounting was collected, but ignored by default. If a system actually ran out of physical swap space and more swap space was immediately needed, the system made the space available by terminating processes with low priority (such as batch jobs). This strategy is call "lazy" accounting. The kernel tunable parameter *availsmem_accounting* could be set to enforce strict swap space

accounting. The drawback of lazy accounting is that when IRIX ran out of physical swap space, processes were terminated before they are complete.

In the current version of IRIX, strict swap space accounting is always in effect, but the ability to add both physical and virtual swap space through ordinary system files allows the administrator to add swap space or to effectively turn off strict swap space accounting, without having to either repartition the disk or reconfigure and reboot the system.

Adding Virtual Swap Space

If your system is experiencing processes being denied stack growth or new processes due to a stated lack of swap space, and you believe that there is adequate physical space, add the following entry to your */etc/fstab* file:

```
/usr/swap swap swap pri=4,vlength=204800 0 0
```

Then give the command:

```
mkfile -v 0b /usr/swap
```

The file (*/usr/swap*) will be zero-length, so you have added only virtual swap space, and no real swap area. Your kernel should then allow more processes to execute. However, when an attempt is made to access more than the system limit, IRIX swaps the largest running program out of memory.

Listing Swap Space With The swap -l Command

To determine how much swap space is already configured in your workstation, use the *swap*(1M) command:

```
swap -l
```

If you are running applications that require the system to swap out programs frequently, you may also want to fine-tune the swap area of the disk used by the operating system. For more information on this process, see “Checking for Excessive Paging and Swapping” on page 199.

Checking Swap Activity With The swap -s Command

The *swap -s* command is a very useful tool for determining if you need to add swap space of some sort. The output of the *swap -s* command looks something like:

```
total: 0 allocated + 64248 reserved = 64248 blocks used, 17400 blocks
available
```

Where the fields displayed are as follows (see the *swap*(1M) reference page for more details):

allocated — The number of 512 bytes blocks allocated to private pages (for example, pages that contain data that is in use.)

reserved — The number of 512 byte blocks currently allocated but not yet marked as private pages (the space has been claimed, but is not yet being used.)

blocks used — The number of 512 byte blocks either allocated or reserved (the total number of allocated and reserved blocks)

blocks available — The number of 512 byte blocks available for future reservation and allocation (the total swap shown by the *swap -l* command less the number of blocks used)

Given the following sample *swap -s* output:

```
total: 0 allocated + 34200 reserved = 34200 blocks used, 47448 blocks
available
```

It can be seen that 0 swap blocks are in use, 34200 have been reserved, but not used, which leaves 47448 blocks available for reservation. So, at this point in time, the system above is not swapping, but the programs running on the system have requested approximately 17 Megabytes of swap space, just in case they will need to grow.

Note: 10000 blocks is approximately equal to 5 Megabytes

Many applications reserve what is known as virtual swap space. That is, they request more memory than they will ever need to grow into. The actual size of the application is the amount of physical system resources that the application is utilizing. The virtual size of the application is the amount of system resources it is utilizing plus the amount of extra resources requested but not in use. This is the case in the above example; space has been reserved, but is not in use.

Negative swap space

Let's look at another example of *swap -s* output:

```
total: 41920 allocated + 58736 reserved = 100656 blocks used, -19400
blocks available
```

It may seem worrisome that the swap space available is a negative number. What this means, though, is that some of the allocated/in use pages are located in main memory (RAM). The *swap -s* output does not take main memory into account. The data that is shown in the negative is actually data that is contained in system memory.

It appears that approximately 20 Megabytes of physical swap space is in use, as shown by the amount of allocated space. Therefore, the system is not out of physical swap space. If there was no more physical swap space, the number of allocated blocks would be very close/the same as to the number of blocks reported by the *swap -l* command. Approximately 30 additional Megabytes of swap space has been requested, shown by the requested field, giving a total of 50 Megabytes requested and/or in use. This appears to leave us with an overrun of 10 Megabytes.

Another way to think of that negative number is that the negative number is the amount of physical swap space minus the number of blocks used (allocated + requested). So, as long as this total is less negative than approximately the amount of physical memory (obtained from the *hinv* command) that you have, you have not overrun your system.

The following example shows *swap -s* output of a system that has most likely come to its swapping limit:

```
total: 76920 allocated + 23736 reserved = 100656 blocks used, -19400
blocks available
```

Notice that the total numbers are the same, but the number of allocated blocks is much higher. If the *swap -l* in this example were to report 81000 blocks of physical swap space on the system, it is easy to see that there are only 4000 physical blocks that are not in use.

If *swap -s* reports a negative number, increase virtual swap when your system is not near its physical limits. This will allow your system to allocate space to those applications that grab more space than they actually need. To do this you can turn on virtual swapping by executing the following commands:

```
su
chkconfig vswap on
/etc/init.d/swap start
```

This will allocate more swap space, or space that can be reserved, but not allocated. Please see the */etc/init.d/swap* file and the *swap*(1M) reference page for more information.

If virtual swapping is already *chkconfig*'d on or if the number of allocated blocks is approaching the number of blocks reported by the *swap -l* command, the only way to remedy the situation would be to add more physical memory or swap space. Please see the *swap*(1M) reference page for more information regarding adding swap space (whether through another disk partition or a swap file).

Increasing Swap Space on a One-Disk System

Suppose you don't have the luxury of a multiple-disk system. This section explains how to increase the size of the swap partition on a single disk. You can increase your available swap space by repartitioning your disk, as described earlier in this chapter, or you can add space with the *swap*(1M) command.

The *swap*(1M) command allows you to designate a portion of any disk partition as additional swap space. You can add swap space at any time and delete the new swap space when you no longer need it. There are several options available with this command, and the command is described completely in the *swap*(1M) reference page, but the most convenient method to use is to specify a normal system file as additional swap space.

To specify a file as additional swap space, you first create an empty file of appropriate size with the *mkfile*(1M) command. For example, if you wish to add 10 megabytes of swap space to your system, and you wish that space to be taken from the */usr* file system, use the following *mkfile* command:

```
mkfile –v 10m /usr/tmp.O/moreswap
```

In this command, the **-v** option directs *mkfile* to be verbose in its output to you, which means that you see the following message as a confirmation that the file has been created:

```
/usr/tmp.O/moreswap 10485760 bytes
```

If you do not specify the **-v** option, *mkfile* does its work silently. The second field in the *mkfile* command is the size of the file. In this case, **10m** specifies a file that is 10 megabytes in size. You can use **b**, **k**, or **m** as a suffix to the **size** argument to indicate that the size number is in bytes, kilobytes, or megabytes, respectively. For example, the following commands all produce files of 10 megabytes:

```
mkfile –v 10485760b /usr/tmp.O/moreswap
```

```
mkfile -v 10240k /usr/tmp.O/moreswap
mkfile -v 10m /usr/tmp.O/moreswap
```

Once your file is created, you can use the *swap* command to add it as additional swap space on the system. When you make your file, be certain that the file resides in the file system from which you wish to take the space. The */usr/tmp*.O directory is a good place to use if you wish to use space from the */usr* file system. Typically */usr* will have more available space than the *root* file system (/). Note, however, that you can also use file systems mounted remotely via NFS. Complete information on using remote mounted file systems for swap space is available in the *swap*(1M) reference page.

To begin using your new file as swap space, give the following command:

```
/sbin/swap -a /usr/tmp.O/moreswap
```

The **-a** option indicates that the named file is to be added as swap space immediately. To check your new swap space, use the command:

```
swap -l
```

This command lists all current swap spaces and their status.

To make your new swap file permanent (automatically added at boot time), add the following line to your */etc/fstab* file:

```
/usr/tmp.O/moreswap    swap    swap   pri=3  0  0
```

Note that if you create a swap file in the */tmp* directory of your *root* file system, the file is removed when the system is booted. The */usr/tmp*.O directory of the */usr* file system is not cleaned at boot time, and is therefore a better choice for the location of swap files. If you wish to create your swap files in the *root* file system, first create a */swap* directory, and then create your swap files within that directory.

Increasing Swap Space on a Multidisk System

Adding more swap space to a multidisk system can be done just as if you were adding space on a single disk system. You can always use the *mkfile*(1) and *swap*(1M) commands to add a swap file to your system. However, if you wish to add dedicated swap space in a new disk partition, follow the instructions below.

To double the default amount of swap space, you can use another disk drive as follows:

```
Partition/slice
    0        Temporary space (mount as /tmp)
    1        Swap space
    6        usr2
```

Note that the operating system continually writes onto the partition that is used as swap space, completely destroying any data that might exist there. Be sure that the swap partition does not overlap any user file system partitions. Verify the size of the swap partition in blocks.

Once you choose a partition, create the file */etc/init.d/addswap* to add this partition permanently as a swap partition. Place a line of the following form in the file:

```
swap –a /dev/dsk/devicename 0 length
```

The argument *devicename* is the device name where the swap partition is located (such as *ips0d1s1*), and *length* is on blocks. Once you create this file, use the *chmod*(1) command to enable execute permission on the file. The command is:

```
chmod +x addswap
```

Next, create a symbolic link to the new file with the command:

```
ln –s /etc/init.d/addswap /etc/rc2.d/S59addswap
```

The */etc/rc2.d* directory controls the system activities that take place when the system boots into multiuser mode (run level 2). The "S" at the beginning of the symbolic linkfile that you created indicates that the commands in the file should be *started* when the system initiates this run level. Symbolic link files that begin with the letter "K" indicate that the commands described in the file should be *killed*. The number following the S or K at the beginning of the linkfile name indicates the sequence in which the commands are executed.

You can also modify the file */etc/fstab* to document (in the form of a comment) that the chosen partition is being used as a swap partition.