#38 Ensuring Maximally Compressed Files

As highlighted in Script #37, most Unix implementations include more than one compression method, but the onus is on the user to figure out which does the best job of compressing a given file. What typically happens is that users learn how to work with just one compression program without ever knowing that they could attain better results with a different one. Making this more confusing is that some files compress better with one algorithm and some with another, and there's no way to know without experimentation.

The logical solution is to have a script that compresses files using each of the tools and then selects the smallest resultant file as the best. That's exactly what `bestcompress` does. By the way, this is one of my favorite scripts in the book.

The Code

```
#!/bin/sh

# bestcompress - Given a file, tries compressing it with all the available
#   compression tools and keeps the compressed file that's smallest, reporting
#   the result to the user.  If '-a' isn't specified, bestcompress skips
#   compressed files in the input stream.

Z="compress"     gz="gzip"     bz="bzip2"
Zout="/tmp/bestcompress.$$.Z"
gzout="/tmp/bestcompress.$$.gz"
bzout="/tmp/bestcompress.$$.bz"
skipcompressed=1
```

```
if [ "$1" = "-a" ] ; then
  skipcompressed=0  ; shift
fi

if [ $# -eq 0 ]; then
  echo "Usage: $0 [-a] file or files to optimally compress" >&2; exit 1
fi

trap "/bin/rm -f $Zout $gzout $bzout" EXIT

for name
do
  if [ ! -f "$name" ] ; then
    echo "$0: file $name not found. Skipped." >&2
    continue
  fi

  if [ "$(echo $name | egrep '(\.Z$|\.gz$|\.bz2$)')" != "" ] ; then
    if [ $skipcompressed -eq 1 ] ; then
      echo "Skipped file ${name}: it's already compressed."
      continue
    else
      echo "Warning: Trying to double-compress $name"
    fi
  fi

  $Z  < "$name" > $Zout  &
  $gz < "$name" > $gzout &
  $bz < "$name" > $bzout &

  wait  # run compressions in parallel for speed. Wait until all are done

  smallest="$(ls -l "$name" $Zout $gzout $bzout | \
     awk '{print $5"="NR}' | sort -n | cut -d= -f2 | head -1)"

  case "$smallest" in
     1 ) echo "No space savings by compressing $name. Left as is."
         ;;
     2 ) echo Best compression is with compress. File renamed ${name}.Z
         mv $Zout "${name}.Z" ; rm -f "$name"
         ;;
     3 ) echo Best compression is with gzip. File renamed ${name}.gz
         mv $gzout "${name}.gz" ; rm -f "$name"
         ;;
     4 ) echo Best compression is with bzip2. File renamed ${name}.bz2
         mv $bzout "${name}.bz2" ; rm -f "$name"
  esac

done

exit 0
```

How It Works

The most interesting line in this script is

```
  smallest="$(ls -l "$name" $Zout $gzout $bzout | \
     awk '{print $5"="NR}' | sort -n | cut -d= -f2 | head -1)"
```

This line has `ls` output the size of each file (the original and the three compressed files, in a known order), chops out just the file sizes with `awk`, sorts these numerically, and ends up with the line number of the smallest resultant file. If the compressed versions are all bigger than the original file, the result is 1, and an appropriate message is output. Otherwise, `smallest` will indicate which of `compress`, `gzip`, or `bzip2` did the best job. Then it's just a matter of moving the appropriate file into the current directory and removing the original file.

Another technique in this script is worth pointing out:

```
$Z  < "$name" > $Zout  &
$gz < "$name" > $gzout &
$bz < "$name" > $bzout &
wait
```

The three compression calls are done in parallel by using the trailing `&` to drop each of them into its own subshell, followed by the call to `wait`, which stops the script until all the calls are completed. On a uniprocessor, this might not offer much performance benefit, but with multiple processors, it should spread the task out and complete quite a bit faster.

Running the Script

This script should be invoked with a list of filenames to compress. If some of them are already compressed and you desire to compress them further, use the `-a` flag; otherwise they'll be skipped.

The Results

The best way to demonstrate this script is with a file that needs to be compressed:

```
$ ls -l alice.txt
-rw-r--r--  1 taylor  staff  154872 Dec  4  2002 alice.txt
```

The script hides the process of compressing the file with each of the three compression tools and instead simply displays the results:

```
$ bestcompress alice.txt
Best compression is with compress. File renamed alice.txt.Z
```

You can see that the file is now quite a bit shorter:

```
$ ls -l alice.txt.Z
-rw-r--r--  1 taylor  wheel  66287 Jul  7 17:31 alice.txt.Z
```