Basic shell commands

(Excerpts from http://www.vectorsite.net/tsshell.html#m4)

1 Useful commands:

```
cat                  # Lists a file or files sequentially.
cd                   # Change directories.
chmod +x             # Set execute permissions.
chmod 666            # Set universal read-write permissions.
cp                   # Copy files.
expr 2 + 2           # Add 2 + 2.
fgrep                # Search for string match.
grep                 # Search for string pattern matches.
grep -v              # Search for no match.
grep -n              # List line numbers of matches.
grep -i              # Ignore case.
grep -l              # Only list file names for a match.
head -5 source.txt   # List first 5 lines.
ls                   # Give a simple listing of files.
mkdir                # Make a directory.
more                 # Displays a file a screenfull at a time.
mv                   # Move or rename files.
paste f1 f2          # Paste files by columns.
pg                   # Variant on "more".
pwd                  # Print working directory.
rm                   # Remove files.
rm -r                # Remove entire directory subtree.
rmdir                # Remove a directory.
sed 's/txt/TXT/g'    # Scan and replace text.
sed 's/txt//'        # Scan and delete text.
sed '/txt/q'         # Scan and then quit.
sort                 # Sort input.
sort +1              # Skip first field in sorting.
sort -n              # Sort numbers.
sort -r              # Sort in reverse order.
sort -u              # Eliminate redundant lines in output.
tail -5 source.txt   # List last 5 lines.
tail +5 source.txt   # List all lines after line 5.
tr '[A-Z]' '[a-z]'   # Translate to lowercase.
tr '[a-z]' '[A-Z]'   # Translate to uppercase.
tr -d '_'            # Delete underscores.
uniq                 # Find unique lines.
wc                   # Word count (characters, words, lines).
wc -w                # Word count only.
wc -l                # Line count.
```

2 Elementary shell capabilities:

```
shvar="Test 1"       # Initialize a shell variable.
echo $shvar          # Display a shell variable.
export shvar         # Allow subshells to use shell variable.
mv $f ${f}2          # Append "2" to file name in shell variable.
$1, $2, $3, ...      # Command-line arguments.
$0                   # Shell-program name.
$#                   # Number of arguments.
$*                   # Complete argument list.
shift 2              # Shift argument variables by 2.
read v               # Read input into variable "v".
. mycmds             # Execute commands in file.
```

3 IF statement:

```
if [ "$1" = "red" ]
then
  echo "Illegal code."
  exit
elif [ "$1" = "blue" ]
then
  echo "Illegal code."
  exit
else
  echo "Access granted."
fi

[ "$shvar" = "red" ]     String comparison, true if match.
[ "$shvar" != "red" ]    String comparison, true if no match.
[ "$shvar" = "" ]        True if null variable.
[ "$shvar" != "" ]       True if not null variable.

[ "$nval" -eq 0 ]        Integer test; true if equal to 0.
[ "$nval" -ge 0 ]        Integer test; true if greater than or equal to 0.
[ "$nval" -gt 0 ]        Integer test; true if greater than 0.
[ "$nval" -le 0 ]        Integer test; true if less than or equal to 0.
[ "$nval" -lt 0 ]        Integer test; true if less than to 0.
[ "$nval" -ne 0 ]        Integer test; true if not equal to 0.

[ -d tmp ]               True if "tmp" is a directory.
[ -f tmp ]               True if "tmp" is an ordinary file.
[ -r tmp ]               True if "tmp" can be read.
[ -s tmp ]               True if "tmp" is nonzero length.
[ -w tmp ]               True if "tmp" can be written.
[ -x tmp ]               True if "tmp" is executable.
```

4 CASE statement:

```
case "$1"
in
  "red")      echo "Illegal code."
              exit;;
  "blue")     echo "Illegal code."
              exit;;
  *)          echo "Access granted.";;
esac
```

5 Loop statements:

```
for nvar in 1 2 3 4 5
do
  echo $nvar
done

for file            # Cycle through command-line arguments.
do
  echo $file
done

while [ "$n" != "Joe" ]     # Or:   until [ "$n" = "Joe" ]
do
  echo "What's your name?"
  read n
  echo $n
done
```

There are "break" and "continue" commands that exit or skip to the end of loops as the need arises.