Creating & Modifying Lists of Filenames

1 Developing Lists of Filenames Using Metacharacters

- Metacharacters include: `*`, `?`, `{ }`, and `[ ]`.
- It is considered an error if a **glob** operation results in no file matches (e.g., `rm *.c` when there are no files in the current directory with a .c suffix)
- When this error occurs within a script, the script will exit with the message "`No match`".
- If this is not what you want, then `set nonomatch`.
- If you ever want to use metacharacters, but don't want them to act as metacharacters, then you can `set noglob`; this turns off globbing.

2 Creating Filename Patterns

- `*` matches any sequence of characters, including the null string.
  - `source*.c` means any filename that begins with the characters "`source`", followed by any sequence of characters that ends in "`.c`".
  - `source*` means any filename that begins with the characters "`source`".
  - `*` matches **every** filename (e.g., `cd; rm -r *` is a favorite among those with suicidal tendencies).
- `?` is like `*`, except that it matches any **single** character.
  - `source?.c` matches `source1.c`, but neither `source11.c` nor `source.c`
- `[ ]` are used to define a **closed set of characters** that can be matched between 2 particular strings.
  - `s[abc]x` matches `sax`, `sbx`, and `scx`.
  - `s[a-e]x` matches the files mentioned about plus `sdx` and `sex`.
  - `s[a-d][w-z][0-4]x` matches: `saw0x`, `saw1x`, ...`sdz4x`.
- `{ }` are used together to specify names to be created from substitution strings that are listed between the `{` and the `}` in the 'pattern'.
  - `echo a{g,h,i}p` produces this response: `agp ahp aip`
  - `echo {a,b,c}{1,2,3}.{c,o}` produces

`a1.c a1.o a2.c a2.o a3.c a3.o b1.c b1.o b2.c b2.o b3.c b3.o c1.c c1.o c2.c c2.o c3.c c3.o`

    (My shell doesn't like spaces in these lists.)

  - `rm *.{tex,aux,log,dvi,blg}` will remove all files with a suffix of the form `.tex`, `.aux`, ..., `.blg` (files constructed by latex).

3 Modifying Filenames

There are several operands that modify the filename to which they are applied:

- `:r`

  If `$file=/usr/peter/source/HW1.c`, then `$file.r` is `/usr/peter/source/HW1`
- :e

  `$file.e` is `c`
- `:h`

  `$file.h` is `/usr/peter/source`
- :t
- `$file.r` is `HW1.c`

4 Creating a Unique Filename

- By using the `$$` shell variable, which is the process number of the C shell, you can create unique filenames.
- This is often useful when you are creating files that will be destroyed before the shell is exited, but need to be **unique**.
- For example, `/tmp/$$CC` will be unique for each invocation of the C shell.