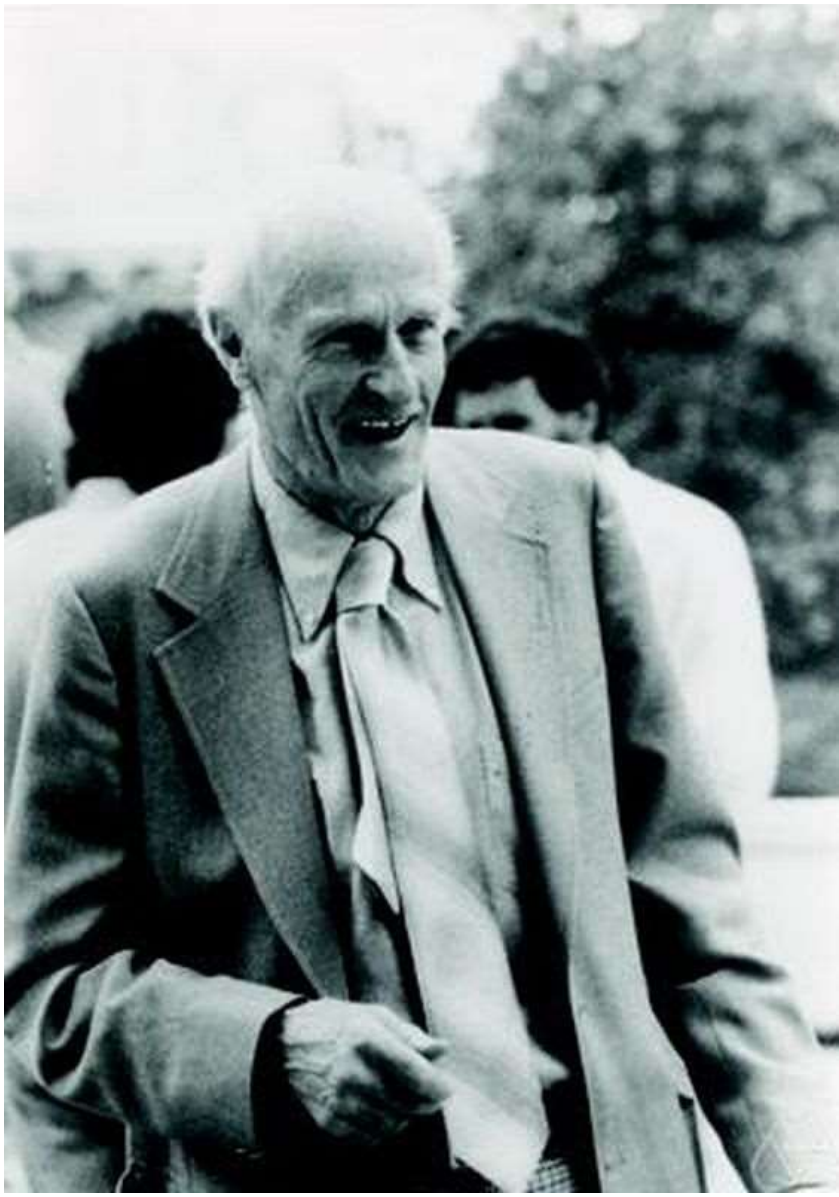
# Regular Expressions in UNIX

Benjamin Brewster

# Regular Expressions

- Regular expressions are a way to specify a pattern of strings that you'd like returned as part of a search

- Invented by Stephen Kleene in the 1950s



# Regular Expressions

- REs are used by many UNIX programs:
  - grep, sed, vi, emacs, regexp, etc.
- Used extensively by many scripting languages:
  - Python, Perl, Tcl/Tk
- There is an entire course (CS321) that goes over REs, and other grammars

# Common UNIX Commands - Filtering with grep

- **grep** - search for an occurrence of a string that matches a pattern

```
$ cat fileToSearch
FINDME first line
second FINDME line
third line FINDME
fourth line FINDM3
fifth line
sixth lFINDMEine
$ grep "FINDME" fileToSearch
FINDME first line
second FINDME line
third line FINDME
sixth lFINDMEine
```

# Another `grep` example

`ps` returns a list of processes

```
$ ps -ef | grep brewsteb
root     29541 3760  0 11:26 ?     00:00:00 sshd: brewsteb [priv]
brewsteb 29543 29541 0 11:26 ?     00:00:00 sshd: brewsteb@pts/1
brewsteb 29544 29543 0 11:26 pts/1 00:00:00 -csh
brewsteb 30737 29544 0 11:44 pts/1 00:00:00 ps -ef
brewsteb 30738 29544 0 11:44 pts/1 00:00:00 grep brewsteb
```

# Basic REs - Operators

***** (asterisk) – Matches 0 or more of the *previous character*

- *Warning – this is different than Windows and UNIX command line usage!*
- Known as the Kleene Star in the regular grammar field

**^** (circumflex) – When placed at the beginning of a RE, indicates the RE must start at the beginning of the string

**$** (dollar sign) – When placed at the end of an RE, matches the end of the string

# The Asterisk – 0 or more

| **Pattern** | **Matches** |
|---|---|
| A* | A or AA or AAA or … |
| Ab* | Ab or Abb or Abbb or … |
| FINDME* | FINDME or FINDMEE or FINDMEEE… |

Note: ***Not*** FINDMEFINDME or …

# Binding to Beginning and End

- Unless you use the ^ and $ operators, a RE will match substrings

| **Pattern** | **Matches** |
|---|---|
| Jon | Will match any string that contains Jon anywhere |
| ^abc | Any string that *starts* with abc |
| XYZ$ | Any string that *ends* with XYZ |
| ^Ben Brewster$ | Any string that matches “Ben Brewster” *exactly* |

# Single Character Matching

- The following operators are available:

| | |
|---|---|
| . | Matches any single character |
| \ | Causes the following special character to simply be itself |
| | Like the period character itself |
| [abc] | Matches *any one* character inside the brackets |
| [^abc] | Matches any character *except any* of the ones inside |

- Any other non-special character matches itself

# Period Example 1

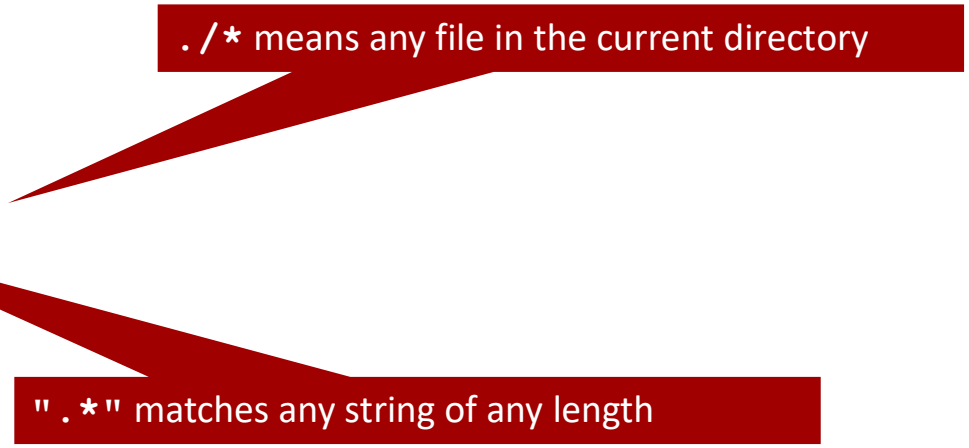
```
$ cat fileToSearch
FINDME first line
second FINDME line
third line FINDME
fourth line FINDM3
fifth line
sixth lFINDMEine
$ grep "FINDM." fileToSearch
FINDME first line
second FINDME line
third line FINDME
fourth line FINDM3
sixth lFINDMEine
```

# Period Example 2 - Two Different Asterisks

.* means match any char any number of times

- This is the “anything, any length” wildcard

```
$ cat datafile
abcdefghi
abcdef
ghi
aabbccddee
$ grep ".*" ./*
abcdefghi
abcdef
ghi
aabbccddee
…
```



# Backslash Example

- The backslash causes the REs to literally interpret special characters

| **Pattern** | **Matches** |
|---|---|
| \. | . |
| \$ | $ |
| * | * |

# Brackets Example 1

```
$ cat fileToSearch
FINDME first line
second FINDME line
third line FINDME
fourth line FINDM3
fifth line
sixth lFINDMEine
$ grep "FINDM[E3]" fileToSearch
FINDME first line
second FINDME line
third line FINDME
fourth line FINDM3
sixth lFINDMEine
```

# Brackets Example 2

```
$ cat fileToSearch
FINDME first line
second FINDME line
third line FINDME
fourth line FINDM3
fifth line
sixth lFINDMEine
$ grep "FINDM[^3]" fileToSearch
FINDME first line
second FINDME line
third line FINDME
sixth lFINDMEine
```

# Brackets Example 3

```
$ cat fileToSearch
FINDME first line
second FINDME line
third line FINDME
fourth line FINDM3
fifth line
sixth lFINDMEine
$ grep "[^3]" fileToSearch
FINDME first line
second FINDME line
third line FINDME
fourth line FINDM3
fifth line
sixth lFINDMEine
```

# Ranges

- When using the square brackets [ ], you can specify ranges of characters to match
- The proper ordering is defined by the ASCII character set:
  - http://www.asciitable.com/

| **Pattern** | **Matches** |
|---|---|
| [a-z] | a b c d e f … y z |
| [^a-z] | Anything but the characters a-z |

# Or: |

```
$ cat catsdogs
i like cats
i like dogs
i like catdogs
i like dogsdogs
$ grep "cat|dog" catsdogs
$ grep "cat\|dog" catsdogs
i like cats
i like dogs
i like catdogs
i like dogsdogs
$ grep "i like \(cat\|dog\)" catsdogs
i like cats
i like dogs
i like catdogs
i like dogsdogs
```

Note that we parenthesize the 'or', here, and that each syntax symbol is escaped

# Matching a Repeated Pattern

- We can search for a pattern that is repeated at least once

```
$ cat catsdogs
i like cats
i like dogs
i like catdogs
i like dogsdogs
$ grep "\(dogs\)\1" catsdogs
i like dogsdogs
```

# Matching a Repeated Pattern

- Curly braces specify the number of repeats (at least) that we're looking for to register a match

```
$ cat digs
dig
digdig
digdigdig
digdigdigdig
digdigdigdigdig

$ grep "\(dig\)\{3\}" digs
digdigdig
digdigdigdig
digdigdigdigdig
```

# Backreferences

`\(  \)` (parentheses) These operators will capture a matched string for later use

`\1, \2, etc.` (escaped integer) This allows you to specify that the string should match the *n*th pattern that you have previously captured, where *n* is the number following the backslash

# Backreference Example

```
$ cat likes
You like You
I like You
$ grep "\(You\) like \1" likes
You like You
```

Backreference

- Note that the repeat example from earlier is actually a backreference:

```
$ grep "\(dogs\)\1" catsdogs
i like dogsdogs
```

Repeat

# Finding Things - `grep`

- Find all lines in all files that match a string:

r :: Search recursively down into each subdirectory
n :: Return the line number of matching lines

Look everywhere from my home directory and deeper

```
$ grep -rn "dogsdogs" ~
/nfs/stak/faculty/b/brewsteb/tempdir/greptests/catsdogs:4:i like dogsdogs
```

# Finding Things - `find` - Example 1

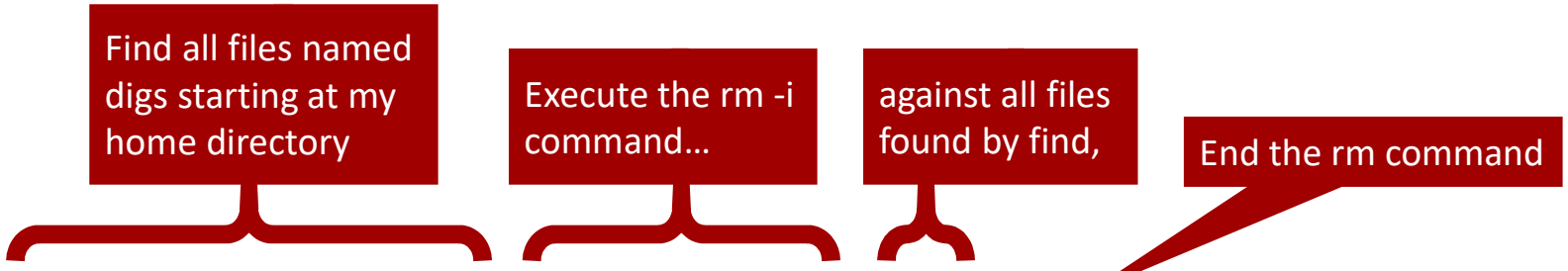
- Find a file by name and many other features
- Can also execute commands against the files found(!)

Find all files named digs starting at my home directory (recursion into directories is implied)

```
$ find ~ -name digs
/nfs/stak/faculty/b/brewsteb/tempdir/greptests/digs
```

# Finding Things - `find` - Example 2

- Find a file by name and many other features
- Can also execute commands against the files found(!)




```
$ find ~ -name digs -exec rm -i '{}' \;
rm: remove regular empty file
'/nfs/stak/faculty/b/brewsteb/tempdir/greptests/digs'? y
```

# Finding Things - `find` - Example 3

- Find a file by name and many other features
- Find based on regular expression

```
$ find ~ -regex ".*fork.*" | sort
/nfs/stak/faculty/b/brewsteb/tempdir/doublefork
/nfs/stak/faculty/b/brewsteb/tempdir/doublefork.c
/nfs/stak/faculty/b/brewsteb/tempdir/forkexec
/nfs/stak/faculty/b/brewsteb/tempdir/forkexec.c
/nfs/stak/faculty/b/brewsteb/tempdir/forkFPsharing
/nfs/stak/faculty/b/brewsteb/tempdir/forkFPsharing.c
/nfs/stak/faculty/b/brewsteb/tempdir/forktest
/nfs/stak/faculty/b/brewsteb/tempdir/forktest.c
/nfs/stak/faculty/b/brewsteb/tempdir/forkwaittest
/nfs/stak/faculty/b/brewsteb/tempdir/forkwaittest.c
/nfs/stak/faculty/b/brewsteb/tempdir/forkyouzombie
/nfs/stak/faculty/b/brewsteb/tempdir/forkyouzombie.c
/nfs/stak/faculty/b/brewsteb/tempdir/pipeNfork
/nfs/stak/faculty/b/brewsteb/tempdir/pipeNfork.c
/nfs/stak/faculty/b/brewsteb/tempdir/pipeNfork_directEdit.c
/nfs/stak/faculty/b/brewsteb/tempdir/pipeNforkFIFO
/nfs/stak/faculty/b/brewsteb/tempdir/pipeNforkFIFO.c
```

# Finding Things - `locate`

- Finds files using a database
- Faster than find, since it doesn't actually search the directory hierarchy for the indicated files
- Will be outdated since last file location check!
- Not available on every system
- Operates differently on those systems that do have it, due to different versions being installed on different Operating Systems
- …just use find, which is ancient and universal