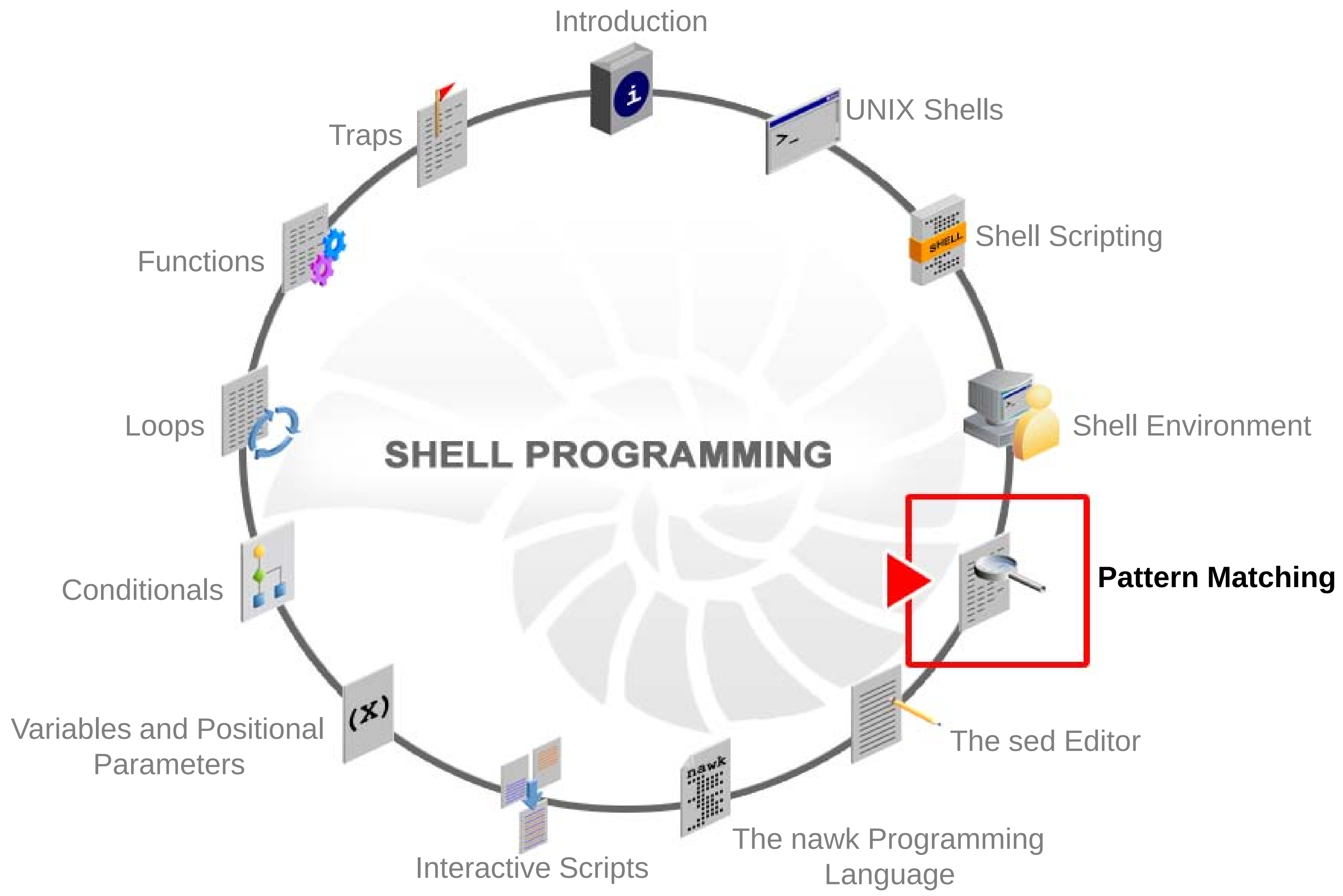
5

Pattern Matching


Introduction
UNIX Shells
Shell Scripting
Shell Environment
Pattern Matching
The sed Editor
The nawk Programming Language
Interactive Scripts
Variables and Positional Parameters
Conditionals
Loops
Functions
Traps
SHELL PROGRAMMING

Objectives

After completing this lesson, you should be able to:

- Find patterns in a file by using the `grep` command
- Explain the role of regular expressions in pattern matching

Agenda

- Finding patterns in a file by using the `grep` command
- Explaining the role of regular expressions in pattern matching

The `grep` Command

- The `grep` command searches one or multiple text files or a pipeline for a specific pattern.
- When the pattern is found, the entire line is printed.
- The command also permits you to use regular expression characters in the search pattern.
- Syntax:

```
grep [OPTIONS] PATTERN [FILE...]
```

- Example:

```
$ ps -ef | grep msxyz
$ ps -e | grep dtterm
352 ?? 0:00 dtterm
353 ?? 0:13 dtterm
354 ?? 0:11 dtterm
1766 pts/5 0:00 dtterm
```

The grep Options

Option	Function
-b	Display the block number at the beginning of each line
-c	Display the number of matched lines
-h	Display the matched lines, but do not display the file names
-i	Ignore case sensitivity
-l	Display the file names, but do not display the matched lines
-n	Display the matched lines and their line numbers
-s	Silent mode
-v	Display all lines that do NOT match
-w	Match whole word

Agenda

- Finding patterns in a file by using the `grep` command
- Explaining the role of regular expressions in pattern matching

Regular Expression

- A regular expression (RE) is a character pattern that matches the same characters in a search.
- Regular expressions:
  - Allow you to specify patterns to search in text
  - Provide a powerful way to search files for specific pattern occurrences
  - Give additional meaning to patterns through metacharacters

Regular Expression Metacharacters

Metacharacter	Function
\	Escapes the special meaning of an RE character
^	Matches the beginning of the line
$	Matches the end of the line
\<	Matches the beginning of the word anchor
\>	Matches the end of the word anchor
[]	Matches any one character from the specified set
[-]	Matches any one character in the specified range
*	Matches zero or more of the preceding character
.	Matches any single character
\{\}	Specifies the minimum and maximum number of matches for a regular expression

Regular Expressions: Example

```
$ ps -ef | grep ’[A-Z]’
ID        PID      PPID     C STIME          TTY      TIME CMD
root      647      1        0 06:14:45       ?        0:00
/usr/lib/dmi/snmpXdmid -s sls-s10-host
noaccess 797      1        0 06:15:03       ?        1:34
/usr/java/bin/java -server -Xmx128m -
XX:+UseParallelGC -XX:ParallelGCThreads=4
root      708      704      4 06:14:50       ?        5:22
/usr/X11/bin/Xorg :0 -depth 24 -nobanner -auth/var/dt/A:0-9Aaayb
root      813      739      0 06:15:16       ?        0:00 /bin/ksh
/usr/dt/bin/Xsession
root      905      903      0 06:15:27       pts/2    0:00 -sh -c unset
DT; DISPLAY=:0; /usr/dt/bin/dtsession_res -merge
root      1045     1        1 06:15:51       ?        1:10
/usr/lib/mixer_applet2 --oaf-
activateiid=OAFIID:GNOME_MixerApplet_Factory --oa
root      1050     1        0 06:15:52       ?        0:01
/usr/lib/notification-area-applet --oafactivate-
iid=OAFIID:GNOME_NotificationA
root      1440     1284     0 08:20:35       pts/4    0:00 grep [A-Z]
<output truncated>
```

Escaping a Regular Expression

- A \ (backslash) character escapes the regular expression characters.
- The \ interprets the next character literally, not as a metacharacter.
- Thus, a `\$` matches a dollar sign and a `\.` matches a period.

Escaping a Regular Expression: Example

```
# grep ’$’ /etc/init.d/nfs.server
#!/sbin/sh
#
# Copyright 2009 Oracle, Inc. All rights reserved.
# Use is subject to license terms.
#
#ident "@(#)nfs.server 1.43 04/07/26 SMI“
# This service is managed by smf(5). Thus, this script provides
# compatibility with previously documented init.d script behaviour.
case "$1" in
'start')
        svcadm enable -t network/nfs/server
        ;;
'stop')
        svcadm disable -t network/nfs/server
        ;;
*)
        echo "Usage: $0 { start | stop }"
        exit 1
        ;;
esac
```

Line Anchors

Line anchors force a regular expression to match only at the start or end of a line.

- Use ^ for the beginning of the line.
- Use $ for the end of the line.

Line Anchors: Example

```
$ grep ’root’ /etc/group
root::0:
other::1:root
bin::2:root,daemon
sys::3:root,bin,adm
adm::4:root,daemon
uucp::5:root

<output truncated>

$ grep ’^root’ /etc/group
root::0:

$ grep ’mount$’ /etc/vfstab
#device device mount FS fsck mount mount
```

Word Anchors

Word anchors are used to refer to the beginning and end of a word in regular expressions.

- Use `\<` for the beginning of the word.
- Use `\>` for the end of the word.

Word Anchors: Example

```
$ grep ’\<uucp’ /etc/group
uucp::5:root

$ grep 'user' /etc/passwd
user:x:100:1::/home/user:/bin/sh
user2:x:101:1::/home/user2:/bin/sh
user3:x:102:1::/home/user3:/bin/sh

$ grep '\<user\>' /etc/passwd
user:x:100:1::/home/user:/bin/sh
```

Character Classes

A character class makes one small sequence of characters match a larger set of characters, such as the following:

- `[abc]` finds a single character in the class.
- `[a-c]` finds a single character in the range.
- `[^a-c]` finds a single character not in the range.

Character Classes: Example

```
$ grep ’[iu]’ /etc/group
bin::2:root,daemon
sys::3:root,bin,adm
uucp::5:root
mail::6:root
nuucp::9:root
sysadmin::14:
nogroup::65534:
$ grep ’[u-y]’ /etc/group
sys::3:root,bin,adm
uucp::5:root
tty::7:root,adm
nuucp::9:root
sysadmin::14:
webservd::80:
nobody::60001:
nogroup::65534:
$ grep ’\<[Tt]he\>’ teams
The teams are chosen randomly.
```

Single Character Match

The `.` (dot) regular expression matches any one character except the newline character.

```
$ grep ’c...h’ /usr/dict/words

$ grep ’^c...h’ /usr/dict/words

$ grep ’^c...h$’ /usr/dict/words
```

Character Match by Specifying a Range

The `\{` and `\}` expressions allow you to specify the minimum and maximum number of matches for a regular expression.

```
$ cat test
root
rooot
roooot
rooooot

$ grep ’ro\{3\}t’ test
rooot

$ grep ’ro\{2,4\}t’ test
root
rooot
roooot
```

Closure Character (*)

- The * symbol, when used in a regular expression, is termed a closure.
- * matches the preceding character zero or more times.

```
$ grep ’Team*’ teams
Team one consists of
Team two consists of
Tea for two and Dom
Tea for two and Tom
$ grep ’\<T.*m\>’ teams
Team one consists of
Tom
Team two consists of
Tea for two and Dom
Tea for two and Tom
$ grep ’*’ teams
$ grep ’abc’ *
data1:abcd
```

The egrep Command

- The `egrep` (extended `grep`) command searches a file or a pipeline for a pattern by using full regular expressions.
- Example:

```
# grep "two | team" teams

# egrep "two | team" teams
Team two consists of
The teams are chosen randomly.
Tea for two and Dom
Tea for two and Tom
```

Quiz

Which of the following forces a regular expression to match only at the end of a line?

a. ^

b. /

c. $

d. *

Summary

In this lesson, you should have learned how to:

- Find patterns in a file by using the `grep` command
- Explain the role of regular expressions in pattern matching

Practice 5 Overview: Pattern Matching

This practice covers the following topics:

- Using Regular Expressions and the `grep` Command
  - You use regular expressions to search for a pattern.
  - You use the `grep` command to search for specific string within text files.