Experiment 6

Exp 6

Theory

awk

The basic function of awk is to search files for lines (or other units of text) that contain certain patterns. When a line matches one of the patterns, awk performs specified actions on that line. awk continues to process input lines in this way until it reaches the end of the input files.

Programs in awk are different from programs in most other languages, because awk programs are data driven (i.e., you describe the data you want to work with and then what to do when you find it). Most other languages are procedural; you have to describe, in great detail, every step the program should take. When working with procedural languages, it is usually much harder to clearly describe the data your program will process. For this reason, awk programs are often refreshingly easy to read and write.

When you run awk, you specify an awk program that tells awk what to do. The program consists of a series of rules (it may also contain function definitions, an advanced feature that we will ignore for now; see section User-Defined Functions). Each rule specifies one pattern to search for and one action to perform upon finding the pattern.

Perl

Perl officially stands for Practical Extraction and Report Language, except when it doesn't.

Perl was originally a language optimized for scanning arbitrary text files, extracting information from those text files, and printing reports based on that information. It quickly became a good language for many system management tasks. Over the years, Perl has grown into a general-purpose programming language. It's widely used for everything from quick "one-liners" to full-scale application development.

The language is intended to be practical (easy to use, efficient, complete) rather than beautiful (tiny, elegant, minimal). It combines (in the author's opinion, anyway) some of the best features of sed, awk, and sh, making it familiar and easy to use for Unix users to whip up quick solutions to annoying problems. Its general-purpose programming facilities support procedural, functional, and object-oriented programming paradigms, making Perl a comfortable language for the long haul on major projects, whatever your bent.

Perl's roots in text processing haven't been forgotten over the years. It still boasts some of the most powerful regular expressions to be found anywhere, and its support for Unicode text is world-class. It handles all kinds of structured text, too, through an extensive collection of extensions. Those libraries, collected in the CPAN, provide ready-made solutions to an astounding array of problems. When they haven't set the standard themselves, they steal from the best -- just like Perl itself.

Grep

grep is one of the most useful and powerful commands in Linux for text processing. grep searches one or more input files for lines that match a regular expression and writes each matching line to standard output.

A regular expression or regex is a pattern that matches a set of strings. A pattern consists of operators, constructs literal characters, and meta-characters, which have special meaning. GNU grep supports three regular expression syntaxes, Basic, Extended, and Perl-compatible.

In its simplest form, when no regular expression type is given, grep interpret search patterns as basic regular expressions. To interpret the pattern as an extended regular expression, use the -E ( or --extended-regexp) option.

In GNU’s implementation of grep there is no functional difference between the basic and extended regular expression syntaxes. The only difference is that in basic regular expressions the meta-characters ?, +, {, |, (, and ) are interpreted as literal characters. To keep the meta-characters' special meanings when using basic regular expressions, the characters must be escaped with a backslash (\). We will explain the meaning of these and other meta-characters later.

Sed

The sed command modifies lines from the specified File parameter according to an edit script and writes them to standard output. The sed command includes many features for selecting lines to be modified and making changes only to the selected lines.

The sed command uses two workspaces for holding the line being modified: the pattern space, where the selected line is held; and the hold space, where a line can be stored temporarily.

An edit script consists of individual subcommands, each one on a separate line. The general form of sed subcommands is the following:

[address-range] function[modifiers]

The sed command processes each input File parameter by reading an input line into a pattern space, applying all sed subcommands in sequence whose addresses select that line, and writing the pattern space to standard output. It then clears the pattern space and repeats this process for each line specified in the input File parameter. Some of the sed subcommands use a hold space to save all or part of the pattern space for subsequent retrieval.

When a command includes an address (either a line number or a search pattern), only the addressed line or lines are affected by the command. Otherwise, the command is applied to all lines.

6a.1

Write a to count number of words in a given file

Output

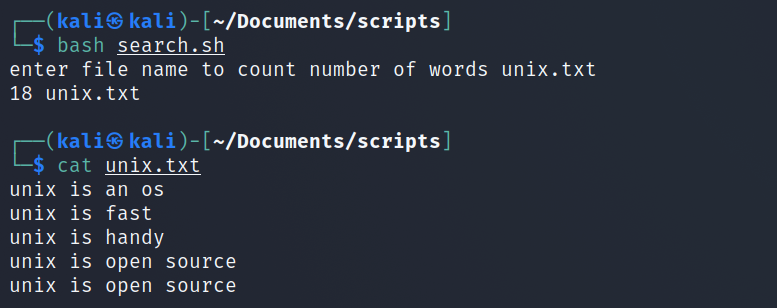


Fig 6a.1.1

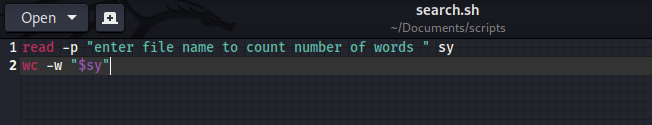


Fig 6a.2.2

6a.2

Write a script using egrep command to display list of specific type of file s in directory

Output

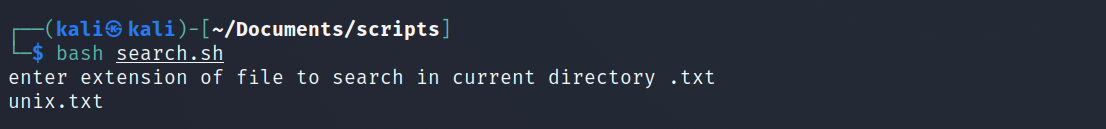


Fig 6.a.2.1

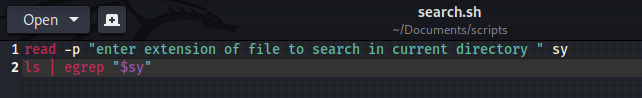


Fig 6.a.2.2s