**1.To insert an element in an array:**

**package** arr;

**public** **class** Arr5 {

/\*\*

\* **@param** args

\*/

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

**int** a[]=**new** **int**[5];//declaration and instantiation

a[0]=10;//initialization

a[1]=20;

a[2]=70;

a[3]=40;

a[4]=50;

//traversing array

**for**(**int** i=0;i<a.length;i++)//length is the property of array

System.*out*.println(a[i]);

}

}

or

**int** a[]={1,2,3,4,5};//declaration and instantiation

2.to find smallest number in an array:

**package** arr;

**public** **class** Arrr2 {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

**int** temp, size;

**int** array[] = {20, 25, 63, 96, 57};

size = array.length;

**for**(**int** i = 0; i<size; i++ ){

**for**(**int** j = i+1; j<size; j++){

**if**(array[i]>array[j]){

temp = array[i];

array[i] = array[j];

array[j] = temp;

}

}

}

System.*out*.println("Smallest element of the array is:: "+array[0]);

}

}

-a show hiden files

Rm r

, is an ansi standard

On is a sequel99 standard

Equi and cross join

Dml are supported in simple view

Only few are “ in complex view

Execute procname b,a

* Match all lines that start with ‘hello’. **E.g:** “hello there”

$ grep “^hello” file1

* Match all lines that end with ‘done’. **E.g:** “well done”

$ grep “done$” file1

* Match all lines that contain any of the letters ‘a’, ‘b’, ‘c’, ‘d’ or ‘e’.

$ grep “[a-e]” file1

* Match all lines that do not contain a vowel

$ grep “[^aeiou]” file1

* Match all lines that start with a digit following zero or more spaces. **E.g:** “ 1.” or “2.”

$ grep “ \*[0-9]” file1

* Match all lines that contain the word hello in upper-case or lower-case

$ grep -i “hello”

**10 UNIX Grep Command Examples of How to Search a File for a Pattern**

The UNIX Grep command searches files for a user-specified text pattern. It returns a list of the matching words or shows each line of text that contains them. You can broaden the results by using wildcards. Grep also has the ability to count instances of a search phrase that appear in a file.   
  
The syntax for this UNIX command includes three parameters, but only the text pattern is required: 

**grep [options] pattern [filename]**

If it contains multiple words or special characters, be sure to place single quotation marks before and after the pattern. The UNIX shell does not interpret dollar signs, brackets, parentheses, pipes or asterisks in the same way as letters. For example, an asterisk is treated as a wildcard when you don't use quotes.   
  
If you leave out the filename, UNIX will collect data from the standard input device. This is usually the keyboard.   
  
**(1) Basic search with no options**   
The following example shows what happens if you use grep to search for the word "unix" in a text file: 

**$ cat examplefile.txt**  
this is line 1 UNIX UNIX  
this is line 2 unix  
this is line 3 Unix Unix  
this is line 4 hello  
  
**$ grep unix examplefile.txt**  
this is line 2 unix

By default, it displays the only line that contains "unix" in lowercase letters.   
  
  
**(2) Ignore upper and lower case distinction**   
One of the grep options allows you to search for words in a case-insensitive manner: 

**$ grep -i unix examplefile.txt**  
this is line 1 UNIX UNIX  
this is line 2 unix  
this is line 3 Unix Unix

All lines containing the word unix, regardless of if it's lowercase, uppercase or a mix of both, are returned.  
  
  
  
  
**(3) Add line numbers to search results**   
The "-n" option adds line numbers to the output: 

**$ grep -n unix examplefile.txt**  
2:this is line 2 unix

This makes it easier to locate the pattern in a large file that doesn't have its own line numbers.   
  
  
**(4) Display how many lines contain the search pattern**   
Another option instructs grep to count the number of times a pattern appears. It will not show any lines or words when you use the "-c" option. 

**$ grep -c unix examplefile.txt**  
1  
  
**$ grep -ci unix examplefile.txt**  
3

The second example above shows that you can always add "-i" to perform case-insensitive searches.   
  
  
**(5) Quiet mode search**   
Adding the -q option prevents grep from writing anything to standard output. This is useful when grep is embedded within a shell script and you want to check if a pattern exists in one or more files but do not want to generate any output during processing. The result (return value) of the search can then be checked by the shell script to determine what logic should be followed. 

**$ grep -q unix examplefile.txt**  
$ echo $?  
0  
  
**$ grep -q bob examplefile.txt**  
$ echo $?  
1

**Note:** It should be obvious, but if it's not 0 indicates true, meaning a match was found. A result of 1 indicates no match was found. If you are not familiar with command return values, or return values in general, 0 typically always means true and a non-zero number (it does not have to be 1) means false.   
  
  
**(6) Show only the part of line matching pattern (Linux ONLY)**   
The grep command in UNIX doesn't support the "-o" option, but you may use it under Linux. It is desirable when you only want to display the matching patterns: 

**$ grep -o unix examplefile.txt**  
unix  
  
**$ grep -oi unix examplefile.txt**  
UNIX  
UNIX  
unix  
Unix  
Unix

Nothing too interesting (or exciting) here.   
  
  
**(7) Show lines from file that DO NOT contain the search pattern**   
You can search for lines that don't contain a pattern by adding the "-v" option. It may be used on both UNIX and Linux systems, and as you will discover, is a VERY handy option to know. 

**$ grep -v unix examplefile.txt**  
this is line 1 UNIX UNIX  
this is line 3 Unix Unix  
this is line 4 hello  
  
**$ grep -vi unix examplefile.txt**  
this is line 4 hello

**(8) Search file for multiple patterns or for patterns beginning with hyphen (-) (Linux ONLY)**   
You may use "-e" to find multiple words or a pattern that begins with a hyphen. The "-e" option is not accepted by grep in UNIX, but Linux will list every line that contains at least one of the words: 

**$ grep -e unix -e hello examplefile.txt**  
this is line 2 unix  
this is line 4 hello

**(9) Search multiple files for a pattern**   
Grep will search multiple files if you add a wildcard to the filename. It lists the name of a file before each line. A wildcard search may come in handy when you can't remember what file contains a certain data item. These UNIX grep command examples show how it works: 

**$ cat examplefile.txt**  
this is line 1 UNIX UNIX  
this is line 2 unix  
this is line 3 Unix Unix  
this is line 4 hello  
  
**$ cat examplefile2.txt**  
this is line 1 hello  
this is line 2 Unix  
this is line 3 UNIX UNIX  
this is line 4 unix unix  
  
**$ grep unix examplefile\***  
examplefile.txt:this is line 2 unix  
examplefile2.txt:this is line 4 unix unix  
  
**$ grep -i unix examplefile\***  
examplefile.txt:this is line 1 UNIX UNIX  
examplefile.txt:this is line 2 unix  
examplefile.txt:this is line 3 Unix Unix  
examplefile2.txt:this is line 2 Unix  
examplefile2.txt:this is line 3 UNIX UNIX  
examplefile2.txt:this is line 4 unix unix

**(10) Recursive search (Linux ONLY)**   
The -r option can be used to recursively search all files under a directory. Although there are multiple methods for achieving the same results (like most things in UNIX and Linux), having an easy to remember option to use with the grep command simplifies the task for you. 

**$ find .**  
.  
./examplefile.txt  
./subdir1  
./subdir1/examplefile2.txt  
./subdir2  
./subdir2/examplefile3.txt  
  
**$ grep -r unix .**  
./examplefile.txt:this is line 2 unix  
./subdir1/examplefile2.txt:this is line 4 unix unix  
./subdir2/examplefile3.txt:this is line 1 unix

Grep has a wide range of practical applications. You may use it to find specific pieces of information in large groups of text files. If you write computer programs, consider using it to generate a list of source code lines that contain a certain variable or function call. The UNIX grep command also proves helpful when you need to determine how many times you have used a particular word in a document.   
  
**Do you need to practice UNIX/Linux commands and writing and running shell scripts...on a REAL SERVER?** If you are ready to move past the basics, either of these online courses is a good place to start...

[UNIX and Linux Operating System Fundamentals](http://www.livefirelabs.com/UNIX_Linux_Fund-tip-4.htm) contains a very good "Introduction to UNIX Shell Scripting" module, and should be taken if you are new to the UNIX and Linux operating system environments or need a refresher on key concepts.  
  
[UNIX Shell Scripting](http://www.livefirelabs.com/Korn_Shell_Scripting-tip-4.htm) is a good option if you are already comfortable with UNIX or Linux and just need to sharpen your knowledge about shell scripting and the UNIX shell in general.

Both courses include access to an Internet Lab system for completing the course's hands-on exercises, which are used to re-enforce the key concepts presented in the course. Any questions you may have while taking the course are answered by an experienced UNIX technologist.