**find and grep**

|  |  |
| --- | --- |
| **find** is a standard Unix utility which searches for files.  find *dirList criteria action*  **find** searches the specified directories and their subdirectories to find files that match the criteria.  The *criteria* can use the following tests:  -name *namePattern* file names matching the *namePattern*  -iname *namePattern* file names matching the *namePattern,* but ignoring case  -user *name* file owner matches this user name or name can be a numeric user ID. | **For examples 1 - 10, assume the following directory tree beginning with the current directory .:**  ./sp1 -> Pgm1/p1  ./Pgm1:  cs2123p1Driver.c cs2123p1.h massign p1abc123.c p1Extra.txt p1OutExtra.txt  cs2123p1Driver.o Makefile p1 p1abc123.o p1Input.txt p1Out.txt  ./Pgm2:  cs2123p2.docx cs2123p2Driver.o Makefile p2abc123.c p2Course.txt  cs2123p2Driver.c cs2123p2.h p2 p2abc123.o p2Query.txt  ./Pgm5:  cs2123p5.h deleteCourse.c insert.o p5Driver.c p6Input.txt  degreePlan.c deleteCourse.o Makefile p5Driver.o printFunctions.c  degreePlan.o insert.c p5 p5Input.txt printFunctions.o  **Example 1**: Find all .o files beginning with the current directory  Why doesn't this work?  $ find . -name \*.o  The shell provides a list of all .o files in the current directory (not subdirectories) as command arguments to find. Since there aren't any, it gives the message:  find: No match  $ find . -name "\*.o"  ./Pgm1/cs2123p1Driver.o  ./Pgm1/p1abc123.o  ./Pgm5/deleteCourse.o  ./Pgm5/degreePlan.o  ./Pgm5/printFunctions.o  ./Pgm5/insert.o  ./Pgm5/p5Driver.o  ./Pgm2/p2abc123.o  ./Pgm2/cs2123p2Driver.o |
| **Finding files based on type or size**  The *criteria* can use the following tests:  -type *fileType* file type matching f (file), d (directory), l (symbolic link)  -size *nSize* file size matches *nSize* where *n:*  +*n* - greater than n  -*n* - less than n  *n* - exactly n  *Size* is  c - characters  k - kilobytes  M - megabytes  G - gigabytes | **Example 2**: Find directories  $ find . -type d  .  ./Pgm1  ./Pgm5  ./Pgm2  **Example 3**: Find files larger than 20 kilobytes.  $ find . -size +20k  ./Pgm5/p5Driver.c  ./Pgm5/p5  ./Pgm2/cs2123p2Driver.o  ./Pgm2/p2  ./Pgm2/cs2123p2Driver.c  **Example 4**: Find symbolic links  $ find . -type l  ./sp1  **Example 5**: Find files smaller than 400 characters  $ find . –size -400c  ./Pgm1/massign  ./Pgm1/Makefile  ./Pgm5/Makefile  ./sp1  ./Pgm2/Makefile |
| **Finding files based on times**  The *criteria* can use the following tests:  -mtime *nTime* file modification time matching *nTime*:  +*n* - more than n days ago  -*n* - less than n days ago  *n* - exactly n days ago  -newer *fileName* if the file being evaluated was modified more recently than the file named *fileName* | **Example 6**: Find files that were modified more than 200 days ago  $ find . -mtime +200  ./Pgm1/cs2123p1Driver.c  ./Pgm1/cs2123p1.h  ./Pgm2/p2Query.txt  ./Pgm2/p2Course.txt  **Example 7**: Find files in directory Pgm5 that were modified after Pgm5/p5Driver.c  $ find Pgm5 -newer Pgm5/p5Driver.c  Pgm5  Pgm5/p5  Pgm5/p5Driver.o |
| **Applying Multiple Criterion with find**  If we list multiple criterion, "and" is assumed by default.  If we want to "or" the criteria, use "-o" between the criterion. | **Example 8**: Modify example 7 to only provide files.  $ find Pgm5 -newer Pgm5/p5Driver.c -type f  Pgm5/p5  Pgm5/p5Driver.o  **Example 9**: Find each .h or .c file  $ find . -name "\*.h" -o -name "\*.c"  ./Pgm1/p1abc123.c  ./Pgm1/cs2123p1Driver.c  ./Pgm1/cs2123p1.h  ./Pgm5/deleteCourse.c  ./Pgm5/printFunctions.c  ./Pgm5/p5Driver.c  ./Pgm5/insert.c  ./Pgm5/degreePlan.c  ./Pgm5/cs2123p5.h  ./Pgm2/cs2123p2.h  ./Pgm2/p2abc123.c  ./Pgm2/cs2123p2Driver.c  For example 9, we could also use:  $ find . -name "\*.[hc]" |
| **Find Actions**  By default, find simply prints the files that are found. We can also send the files directly to a command.  find *dirList criteria* -exec *command* {}\;  For each file matching the criteria, the file is sent to the command. The "{}" are replaced with the file by the find command. | **Example 10**: Find each .h or .c file and list the details using ls.  $ find . -name "\*.h" -o -name "\*.c" -exec ls -al {} \;  -rw------- 1 clark faculty 3403 Jan 6 2017 ./Pgm1/p1abc123.c  -rw------- 1 clark faculty 13030 Dec 19 2016 ./Pgm1/cs2123p1Driver.c  -rw------- 1 clark faculty 2903 Jan 18 2017 ./Pgm5/deleteCourse.c  -rw------- 1 clark faculty 3062 Jan 18 2017 ./Pgm5/printFunctions.c  -rw------- 1 clark faculty 21408 Mar 30 18:27 ./Pgm5/p5Driver.c  -rw------- 1 clark faculty 1949 Jan 18 2017 ./Pgm5/insert.c  -rw------- 1 clark faculty 4839 Jan 18 2017 ./Pgm5/degreePlan.c  -rw------- 1 clark faculty 9840 Jan 7 2017 ./Pgm2/p2abc123.c  -rw------- 1 clark faculty 28407 Jan 7 2017 ./Pgm2/cs2123p2Driver.c |
| **grep** is a standard Unix utility the searches for text strings within files:  grep *options pattern fileList*  grep *options pattern*  grep searches *fileList* for files that match the specified pattern which can be a regular expression. Without *fileList,* grep receives text from stdin.  The *fileList* is a list of files that grep searches. The shell provides the files.  The *pattern* can be a simple string or a regular expression.  Major options affecting the *pattern*:  -F fixed pattern (i.e., isn't a regular expression)  -G regular grep pattern (i.e., regular expression)  -E extended regular expression pattern  Create a GrepExamples directory. When logged into a fox server, please cd to the /usr/local/courses/clark/cs3423/GrepExamples directory and copy all the files to your GrepExamples directory. | **Example 11-1**: Find text lines containing "cat" in the GrepExamples directory. The -F options specifies that we want a fixed string (not a regex) and causes grep to execute more efficiently.  $ grep -F "cat" \*  file4:John bought a cat, but it was really a very smelly cat.  file4:He had hoped to get a fun cat.  file5:The weather was getting very bad. It was raining cats and dogs.  file5:what a catastrophe.  file5:I was so tired that I needed a cat nap.  grep: Program2: Is a directory  grep: Program3: Is a directory  grep: Program4: Is a directory  grep: Program5: Is a directory  **Example 11-2**: Find text lines containing "cat" in the GrepExamples directory and its subdirectories. The -r option recurses to subdirectories.  $ grep -F -r "cat" \*  file4:John bought a cat, but it was really a very smelly cat.  file4:He had hoped to get a fun cat.  file5:The weather was getting very bad. It was raining cats and dogs.  file5:what a catastrophe.  file5:I was so tired that I needed a cat nap.  Program3/cs1713p3Driver.c: specifies the beginning of customer request and Program3/cs1713p3Driver.c: - If the token is larger than the szToken parm, we return Program3/cs1713p3Driver.c: iCopy = MAX\_TOKEN\_SIZE; // truncated size  Program4/cs1713p4.h:Node \*allocateNode(Book book);  Program4/cs1713p4Driver.c: specifies the beginning of customer request and includes Program4/cs1713p4Driver.c:/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* allocateNode \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Program4/cs1713p4Driver.c: Node \* allocateNode(Book book)  Program4/cs1713p4Driver.c: Allocates a new node, placing the parameter in the node.  Program4/cs1713p4Driver.c: A pointer to the newly allocated node.  Program4/cs1713p4Driver.c:Node \* allocateNode(Book book)  Program4/cs1713p4Driver.c: // to allocate a new node  Program4/cs1713p4Driver.c: exitError("Memory allocation error", "");  Program4/cs1713p4Driver.c: - If the token is larger than the szToken parm, we return a  Program4/cs1713p4Driver.c: iCopy = MAX\_TOKEN\_SIZE; // truncated size  Program5/cs1713p5Driver.c: specifies the beginning of customer request and includes Program5/cs1713p5Driver.c:/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* allocateNodeT \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Program5/cs1713p5Driver.c: NodeT \* allocateNodeT(Book book)  Program5/cs1713p5Driver.c: Allocates a new node, placing the parameter in the node.  Program5/cs1713p5Driver.c: A pointer to the newly allocated node.  Program5/cs1713p5Driver.c:NodeT \*allocateNodeT(Book book)  Program5/cs1713p5Driver.c: // to allocate a new node  Program5/cs1713p5Driver.c: exitError("Memory allocation error", "");  Program5/cs1713p5Driver.c: - If the token is larger than the szToken parm, we return a  Program5/cs1713p5Driver.c: iCopy = MAX\_TOKEN\_SIZE; // truncated size  Program5/cs1713p5.h:NodeT \*allocateNodeT(Book book); |
| **Important Grep Options**  -r recurse to subdirectories  -c count occurrences in each file  -w matches as a whole word not as a substring of another word | **Example 12**: Count the text lines in each file that contain "cat".  $ grep -F -r -c "cat" \*  cs1713p0.c:0  cs1713p0v2.c:0  file4:2  file5:3  file6:0  Program2/p2Book.txt:0  Program2/cs1713p2.h:0  Program2/cs1713p2Stuff.c:0  Program2/p2Out.txt:0  Program2/p2Customer.txt:0  Program3/p3Book.txt:0  Program3/cs1713p3.h:0  Program3/cs1713p3Driver.c:3  Program3/p3Command.txt:0  Program4/p4Book.txt:0  Program4/cs1713p4.h:1  Program4/cs1713p4Driver.c:10  Program4/p4Command.txt:0  Program4/p4Commandv2.txt:0  Program5/p5Book.txt:0  Program5/Makefile:0  Program5/cs1713p5Driver.c:10  Program5/p5Command.txt:0  Program5/cs1713p5.h:1  **Example 13**: Show text lines that only match the entire word "cat", but not as a substring of another word.  $ grep -F -r -w "cat" \*  file4:John bought a cat, but it was really a very smelly cat.  file4:He had hoped to get a fun cat.  file5:I was so tired that I needed a cat nap. |
| **Important Grep Options (continued)**  -n display the line numbers for any matched text  -h don't print the file name | **Example 14**: Repeat example 13, but also show the line numbers  $ grep -F -r -w -n "cat" \*  file4:1:John bought a cat, but it was really a very smelly cat.  file4:2:He had hoped to get a fun cat.  file5:4:I was so tired that I needed a cat nap.  **Example 15**: Show the lines containing #include statements sorted by include file name. Don't show the filenames.  $ grep –F –r -h “#include” \* | sort  #include "cs1713p0.h"  #include "cs1713p0.h"  #include "cs1713p3.h"  #include "cs1713p4.h"  #include "cs1713p5.h"  #include <stdio.h>  #include <stdio.h>  #include <stdio.h>  #include <stdio.h>  #include <stdio.h>  #include <stdlib.h>  #include <stdlib.h>  #include <stdlib.h>  #include <string.h>  #include <string.h>  #include <string.h> |
| **Another Tinker Toy Example**  Use the **uniq -c** command which counts the number of occurrences of each unique line. | **Example 16**: Show the lines containing #include statements sorted by include file name. Don't show the filenames.  $ grep –F –r -h | sort | uniq -c  2 #include "cs1713p0.h"  1 #include "cs1713p3.h"  1 #include "cs1713p4.h"  1 #include "cs1713p5.h"  5 #include <stdio.h>  3 #include <stdlib.h>  3 #include <string.h> |
| **Important Grep Options (continued)**  -f compare a file of patterns against the *fileList*  grep -f *patternFile fileList*  Each line in the *patternFile* represents a pattern to be matched. The patterns can be regular expressions. | **Example 17**: We have a pattern file that contains subroutines we want to find in text files.  $ vi example17pat.txt  printf  sscanf  strcpy  $ grep -f example17pat.txt cs1713p0.c  // about the safety of scanf and printf  int iScanfCnt; // sscanf returns the number of successful inputs  printf("%-10s %-20s %10s %10s %10s %10s\n"  iScanfCnt = sscanf(szInputBuffer, "%lf %lf %lf %6s %20[^\n]\n"  printf("invalid input when reading student data, only %d valid values. \n"  printf("\tdata is %s\n", szInputBuffer);  printf("%-10s %-20s %10.2f %10.2f %10.2f %10.2f\n"  **How could we eliminate the lines having "sscanf" or "printf" which aren't function calls?**  $ vi example17pat.txt  printf(  sscanf(  strcpy |
| **Important Grep Options (continued)**  -l when matching, only list the file names in the result instead of also showing the text line  -d skip don't recurse into subdirectories | **Example 18:** in bash, use grep to get a list of files that contain the first argument. Use cat to show the contents of those files.  $ vi example18.bash  #!/bin/bash  # Use grep to get a list of files that contain the first argument.  # Use cat to show the contents of those files.  for file in $(grep -d skip -l $1 \*); do  echo "$file:"  cat <$file  done  $ bash example18.bash cat  example18.bash:  #!/bin/bash  # Use grep to get a list of files that contain the first argument.  # Use cat to show the contents of those files.  for file in $(grep -d skip -l $1 \*); do  echo "$file:"  cat <$file  done  file4:  John bought a cat, but it was really a very smelly cat.  He had hoped to get a fun cat.  file5:  The weather was getting very bad. It was raining cats and dogs.  It was so bad that I stepped in a poodle.  what a catastrophe.  I was so tired that I needed a cat nap. |
| **Instead of discussing regular expressions in detail here, we plan to discuss them in detail with sed and Perl.** |  |

**©2017 Larry W. Clark,** UTSA CS students may make copies for their personal use