Sinatra Cheat Sheet Project:

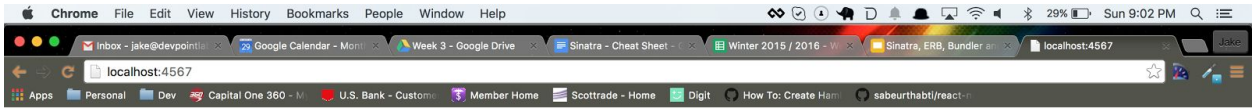
Basic Objectives:

1. Create a new directory for your Sinatra project
2. Create a new ruby file for your Sinatra server
3. Have links on your root page to show hardcoded man files for the cp, ls, mkdir, touch and mv commands.
4. Have a link to go back to the root page after seeing a man page for a specific command

Bonus objectives:

1. Have a menu with links
    a. Have a menu (*see example pictures below)*
2. Have a search form as one of the menu items
3. Submit the form (POST) to your server and get a response of the man page back or display “Nothing was found” if the man command returns an empty string
4. Have links to go to and from your menus
5. Have a link on the main menu that goes to google for a google search
    a. this link should open in a new tab so it doesn’t disturb the user's command line tab
6. Style the application with CSS
    a. remember inline css is bad, in file css is pretty bad, external file css is best (hint: you’ll probably have to google how to do this. We are only going to cover the very basics of Sinatra and then go right into the Rails Web Framework)
7. Use partials to clean up your views
    a. We may or may not talk about these in sinatra depending on how much time we have in class and where the class is at. (hint: you’ll probably have to google how to do this. We are only going to cover the very basics of Sinatra and then go right into the Rails Web Framework)

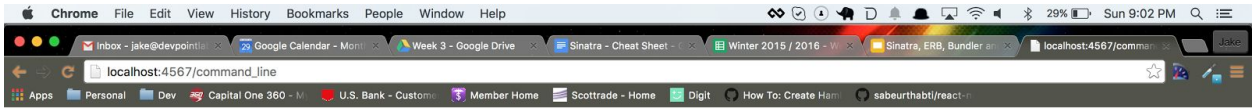
SCROLL DOWN FOR SOME BASIC EXAMPLE SCREENSHOTS



Command Line Menu

---

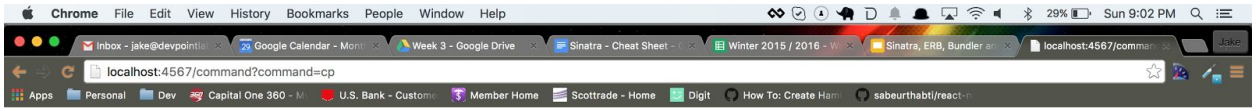
1. Command Line
2. Search
3. Google



COMMAND LINE

1. CP
2. LS
3. MKDIR
4. TOUCH
5. MV

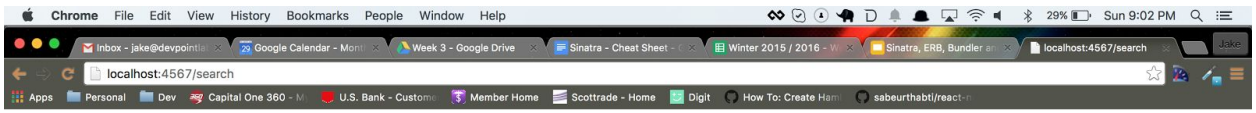
Menu



COMMAND RESULT

Menu Commands

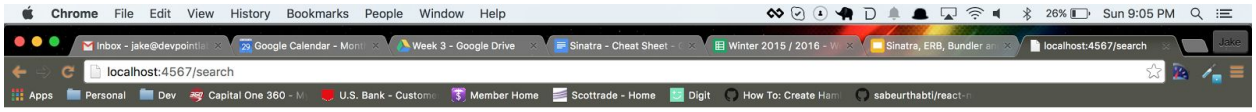
CP(1) BSD General Commands Manual CP(1) NAME cp -- copy files SYNOPSIS cp [-R [-H | -L | -P]] [-fi | -n] [-apvX] source_file target_file cp [-R [-H | -L | -P]] [-fi | -n] [-apvX] source_file ... target_directory DESCRIPTION In the first synopsis form, the cp utility copies the contents of the source_file to the target_file. In the second synopsis form, the con- tents of each named source_file is copied to the destination target_directory. The names of the files themselves are not changed. If cp detects an attempt to copy a file to itself, the copy will fail. The following options are available: -a Same as -pPR options. Preserves structure and attributes of files but not directory structure. -f If the destination file cannot be opened, remove it and create a new file, without prompting for confirmation regardless of its per- missions. (The -f option overrides any previous -n option.) The target file is not unlinked before the copy. Thus, any exist- ing access rights will be retained. -H If the -R option is specified, symbolic links on the command line are followed. (Symbolic links encountered in the tree traversal are not followed.) -i Cause cp to write a prompt to the standard error output before copying a file that would overwrite an existing file. If the response from the standard input begins with the character `y' or `Y', the file copy is attempted. (The -i option overrides any pre- vious -n option.) -L If the -R option is specified, all symbolic links are followed. -n Do not overwrite an existing file. (The -n option overrides any previous -f or -i options.) -P If the -R option is specified, no symbolic links are followed. This is the default. -p Cause cp to preserve the following attributes of each source file in the copy: modification time, access time, file flags, file mode, user ID, and group ID, as allowed by permissions. Access Control Lists (ACLs) and Extended Attributes (EAs), including resource forks, will also be preserved. If the user ID and group ID cannot be preserved, no error message is displayed and the exit value is not altered. If the source file has its set-user-ID bit on and the user ID can- not be preserved, the set-user-ID bit is not preserved in the copy's permissions. If the source file has its set-group-ID bit on and the group ID cannot be preserved, the set-group-ID bit is not preserved in the copy's permissions. If the source file has both its set-user-ID and set-group-ID bits on, and either the user ID or group ID cannot be preserved, neither the set-user-ID nor set- group-ID bits are preserved in the copy's permissions. -R If source_file designates a directory, cp copies the directory and the entire subtree connected at that point. If the source_file ends in a /, the contents of the directory are copied rather than the directory itself. This option also causes symbolic links to be copied, rather than indirected through, and for cp to create spe- cial files rather than copying them as normal files. Created directories have the same mode as the corresponding source direc- tory, unmodified by the process' umask. In -R mode, cp will continue copying even if errors are detected. Note that cp copies hard-linked files as separate files. If you need to preserve hard links, consider using tar(1), cpio(1), or pax(1) instead. -v Cause cp to be verbose, showing files as they are copied. -X Do not copy Extended Attributes (EAs) or resource forks. For each destination file that already exists, its contents are overwrit- ten if permissions allow. Its mode, user ID, and group ID are unchanged unless the -p option was specified. In the second synopsis form, target_directory must exist unless there is only one named source_file which is a directory and the -R flag is speci- fied. If the destination file does not exist, the mode of the source file is used as modified by the file mode creation mask (umask, see csh(1)). If the source file has its set-user-ID bit on, that bit is removed unless both the source file and the destination file are owned by the same user. If the source file has its set-group-ID bit on, that bit is removed unless both the source file and the destination file are in the same group and the user is a member of that group. If both the set-user-ID and set-group-ID bits are set, all of the above conditions must be ful- filled or both bits are removed. Appropriate permissions are required for file creation or overwriting. Symbolic links are always followed unless the -R flag is set, in which case symbolic links are not followed, by default. The -H or -L flags (in conjunction with the -R flag) cause symbolic links to be followed as described above. The -H, -L and -P options are ignored unless the -R option is specified. In addition, these options override each other and the command's actions are determined by the last one specified. If cp receives a SIGINFO (see the status argument for stty(1)) signal, the current input and output file and the percentage complete will be written to the standard output. EXIT STATUS The cp utility exits 0 on success, and >0 if an error occurs. COMPATIBILITY Historic versions of the cp utility had a -r option. This implementation supports that option; however, its use is strongly discouraged, as it does not correctly copy special files, symbolic links, or fifo's. The -v and -n options are non-standard and their use in scripts is not recommended. LEGACY DESCRIPTION In legacy mode, -f will override -i. Also, under the -f option, the tar- get file is always unlinked before the copy. Thus, new access rights will always be set. In -R mode, copying will terminate if an error is encountered. For more information about legacy mode, see compat(5). SEE ALSO mv(1), rcp(1), umask(2), fts(3), compat(5), symlink(7) STANDARDS The cp command is expected to be IEEE Std 1003.2 (``POSIX.2'') compati- ble. HISTORY A cp command appeared in Version 1 AT&T UNIX. BSD February 23, 2005 BSD



MAN SEARCH

Submit

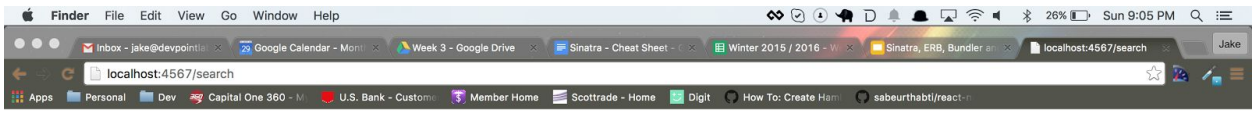
Menu



MAN SEARCH

vim Submit

Menu



RESULT

Menu

VIM(1) VIM(1) NAME vim - Vi IMproved, a programmers text editor SYNOPSIS vim [options] [file ..] vim [options] - vim [options] -t tag vim [options] -q [errorfile] ex view gvim gview evim eview rvim rview rgvim rgview DESCRIPTION Vim is a text editor that is upwards compatible to Vi. It can be used to edit all kinds of plain text. It is especially useful for editing programs. There are a lot of enhancements above Vi: multi level undo, multi win- dows and buffers, syntax highlighting, command line editing, filename completion, on-line help, visual selection, etc.. See ":help vi_diff.txt" for a summary of the differences between Vim and Vi. While running Vim a lot of help can be obtained from the on-line help system, with the ":help" command. See the ON-LINE HELP section below. Most often Vim is started to edit a single file with the command vim file More generally Vim is started with: vim [options] [filelist] If the filelist is missing, the editor will start with an empty buffer. Otherwise exactly one out of the following four may be used to choose one or more files to be edited. file .. A list of filenames. The first one will be the current file and read into the buffer. The cursor will be posi- tioned on the first line of the buffer. You can get to the other files with the ":next" command. To edit a file that starts with a dash, precede the filelist with "--". - The file to edit is read from stdin. Commands are read from stderr, which should be a tty. -t {tag} The file to edit and the initial cursor position depends on a "tag", a sort of goto label. {tag} is looked up in the tags file, the associated file becomes the current file and the associated command is executed. Mostly this is used for C programs, in which case {tag} could be a function name. The effect is that the file containing that function becomes the current file and the cursor is positioned on the start of the function. See ":help tag-commands". -q [errorfile] Start in quickFix mode. The file [errorfile] is read and the first error is displayed. If [errorfile] is omitted, the filename is obtained from the 'errorfile' option (defaults to "AztecC.Err" for the Amiga, "errors.err" on other systems). Further errors can be jumped to with the ":cn" command. See ":help quickfix". Vim behaves differently, depending on the name of the command (the exe- cutable may still be the same file). vim The "normal" way, everything is default. ex Start in Ex mode. Go to Normal mode with the ":vi" command. Can also be done with the "-e" argument. view Start in read-only mode. You will be protected from writing the files. Can also be done with the "-R" argument. gvim gview The GUI version. Starts a new window. Can also be done with the "-g" argument. evim eview The GUI version in easy mode. Starts a new window. Can also be done with the "-y" argument. rvim rview rgvim rgview Like the above, but with restrictions. It will not be possi- ble to start shell commands, or suspend Vim. Can also be done with the "-Z" argument. OPTIONS The options may be given in any order, before or after filenames. Options without an argument can be combined after a single dash. +[num] For the first file the cursor will be positioned on line "num". If "num" is missing, the cursor will be positioned on the last line. +/{pat} For the first file the cursor will be positioned on the first occurrence of {pat}. See ":help search-pattern" for the available search patterns. +{command} -c {command} {command} will be executed after the first file has been read. {command} is interpreted as an Ex command. If the {command} contains spaces it must be enclosed in double quotes (this depends on the shell that is used). Example: Vim "+set si" main.c Note: You can use up to 10 "+" or "-c" commands. -S {file} {file} will be sourced after the first file has been read. This is equivalent to -c "source {file}". {file} cannot start with '-'. If {file} is omitted "Session.vim" is used (only works when -S is the last argument). --cmd {command} Like using "-c", but the command is executed just before processing any vimrc file. You can use up to 10 of these commands, independently from "-c" commands. -A If Vim has been compiled with ARABIC support for editing right-to-left oriented files and Arabic keyboard mapping, this option starts Vim in Arabic mode, i.e. 'arabic' is set. Otherwise an error message is given and Vim aborts. -b Binary mode. A few options will be set that makes it pos- sible to edit a binary or executable file. -C Compatible. Set the 'compatible' option. This will make Vim behave mostly like Vi, even though a .vimrc file exists. -d Start in diff mode. There should be two, three or four file name arguments. Vim will open all the files and show differences between them. Works like vimdiff(1). -d {device} Open {device} for use as a terminal. Only on the Amiga. Example: "-d con:20/30/600/150". -D Debugging. Go to debugging mode when executing the first command from a script. -e Start Vim in Ex mode, just like the executable was called "ex". -E Start Vim in improved Ex mode, just like the executable was called "exim". -f Foreground. For the GUI version, Vim will not fork and detach from the shell it was started in. On the Amiga, Vim is not restarted to open a new window. This option should be used when Vim is executed by a program that will wait for the edit session to finish (e.g. mail). On the Amiga the ":sh" and ":!" commands will not work. --nofork Foreground. For the GUI version, Vim will not fork and detach from the shell it was started in. -F If Vim has been compiled with FKMAP support for editing right-to-left oriented files and Farsi keyboard mapping, this option starts Vim in Farsi mode, i.e. 'fkmap' and 'rightleft' are set. Otherwise an error message is given and Vim aborts. -g If Vim has been compiled with GUI support, this option enables the GUI. If no GUI support was compiled in, an error message is given and Vim aborts. -h Give a bit of help about the command line arguments and options. After this Vim exits. -H If Vim has been compiled with RIGHTLEFT support for editing right-to-left oriented files and Hebrew keyboard mapping, this option starts Vim in Hebrew mode, i.e. 'hkmap' and 'rightleft' are set. Otherwise an error message is given and Vim aborts. -i {viminfo} When using the viminfo file is enabled, this option sets the filename to use, instead of the default "~/.viminfo". This can also be used to skip the use of the .viminfo file, by giving the name "NONE". -L Same as -r. -l Lisp mode. Sets the 'lisp' and 'showmatch' options on. -m Modifying files is disabled. Resets the 'write' option. You can still modify the buffer, but writing a file is not possible. -M Modifications not allowed. The 'modifiable' and 'write' options will be unset, so that changes are not allowed and files can not be written. Note that these options can be set to enable making modifications. -N No-compatible mode. Reset the 'compatible' option. This will make Vim behave a bit better, but less Vi compatible, even though a .vimrc file does not exist. -n No swap file will be used. Recovery after a crash will be impossible. Handy if you want to edit a file on a very slow medium (e.g. floppy). Can also be done with ":set uc=0". Can be undone with ":set uc=200". -nb Become an editor server for NetBeans. See the docs for details. -o[N] Open N windows stacked. When N is omitted, open one window for each file. -O[N] Open N windows side by side. When N is omitted, open one window for each file. -p[N] Open N tab pages. When N is omitted, open one tab page for each file. -R Read-only mode. The 'readonly' option will be set. You can still edit the buffer, but will be prevented from acci- dently overwriting a file. If you do want to overwrite a file, add an exclamation mark to the Ex command, as in ":w!". The -R option also implies the -n option (see below). The 'readonly' option can be reset with ":set noro". See ":help 'readonly'". -r List swap files, with information about using them for recovery. -r {file} Recovery mode. The swap file is used to recover a crashed editing session. The swap file is a file with the same filename as the text file with ".swp" appended. See ":help recovery". -s Silent mode. Only when started as "Ex" or when the "-e" option was given before the "-s" option. -s {scriptin}