2. Study of a terminal based text editor such as Vim or Emacs. (By the end of the course, students are expected to acquire following skills in using the editor: cursor operations, manipulate text, search for patterns, global search and replace) Basic Linux commands, familiarity with following commands/operations expected

1. man

2. ls, echo, read

3. more, less, cat,

4. cd, mkdir, pwd, find

5. mv, cp, rm ,tar

6. wc, cut, paste

7. head, tail, grep, expr

8 chmod, chown

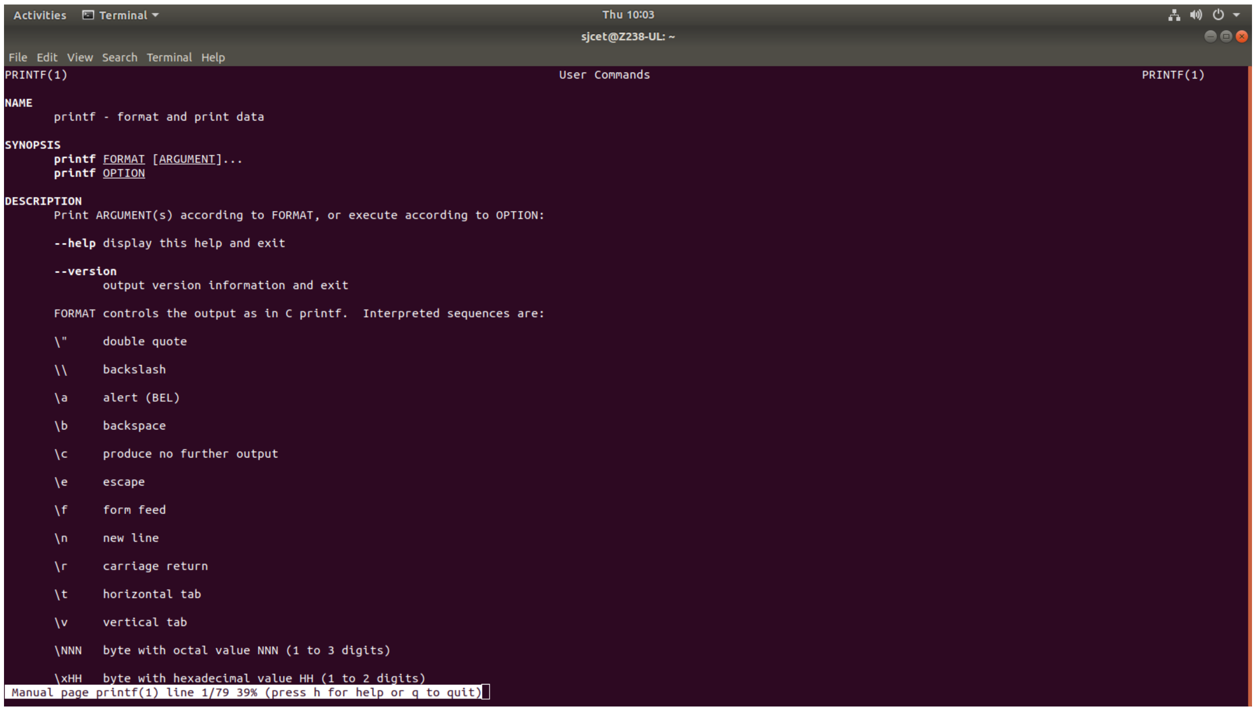
9. Redirections & Piping

10. useradd, usermod, userdel, passwd

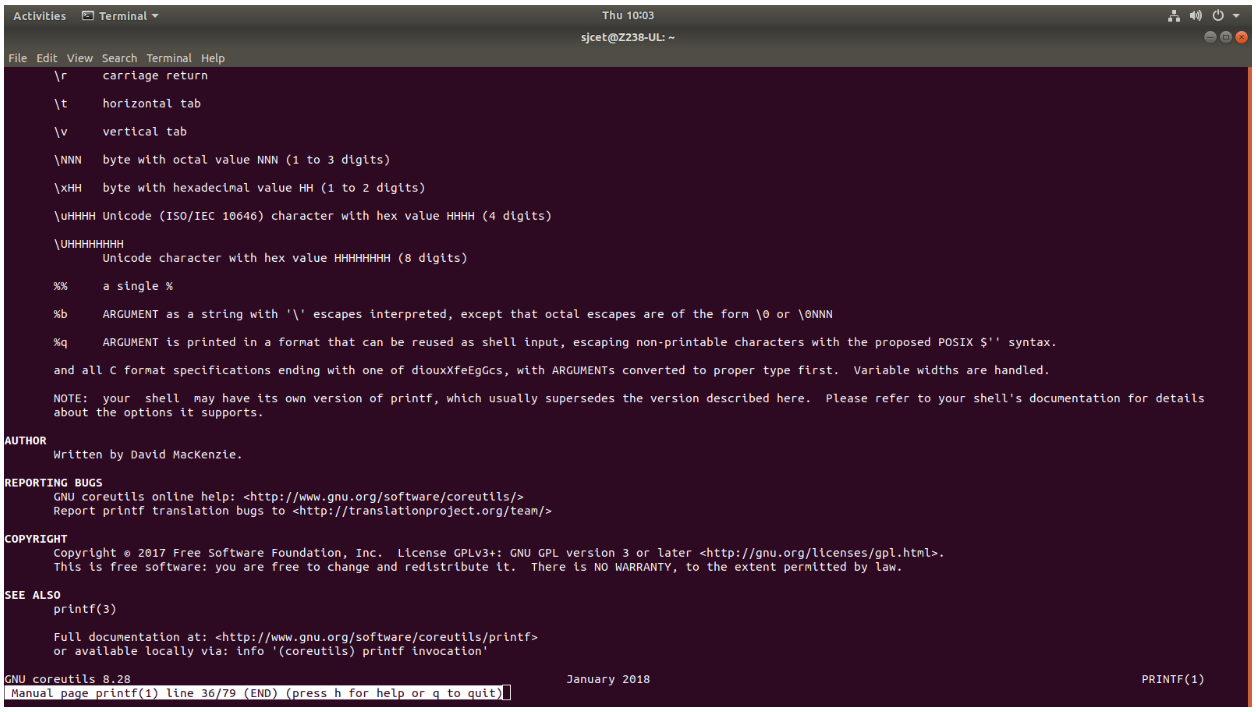
11. df,top, ps

12 ssh, scp, ssh-keygen, ssh-copy-id.

1_cmd: `$ man printf`

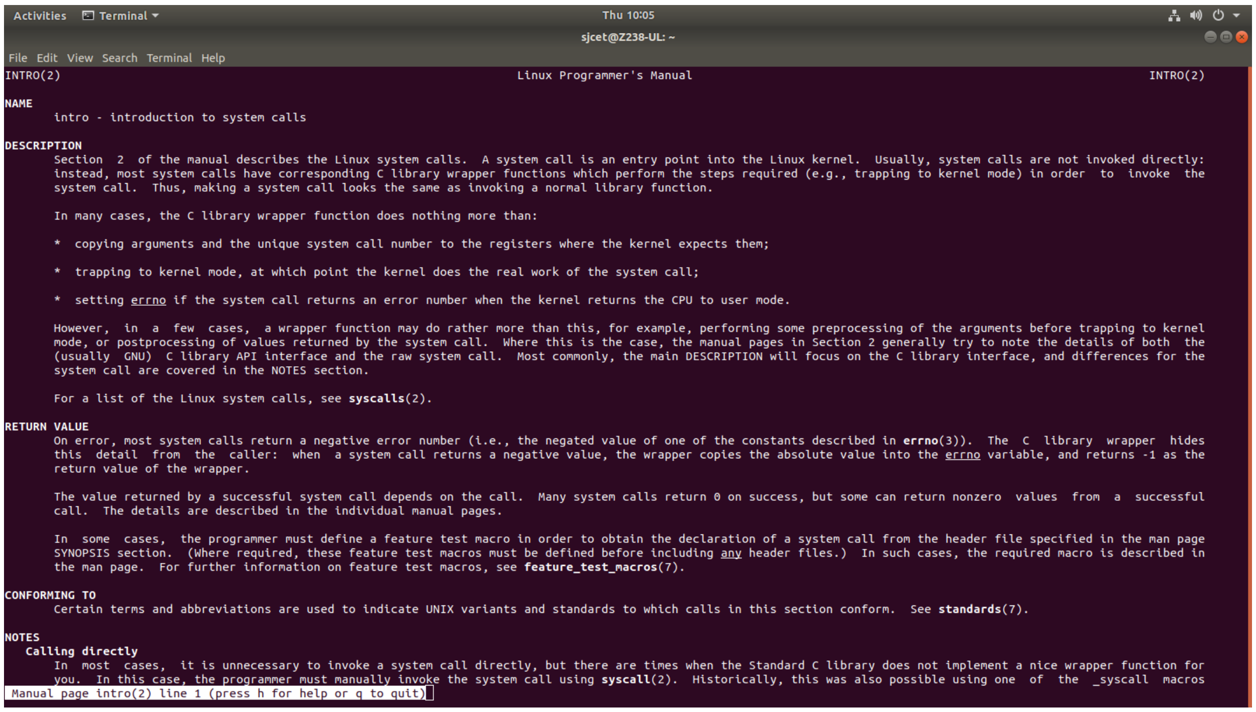


```
PRINTF(1)                                        User Commands                                        PRINTF(1)

NAME
       printf - format and print data

SYNOPSIS
       printf FORMAT [ARGUMENT]...
       printf OPTION

DESCRIPTION
       Print ARGUMENT(s) according to FORMAT, or execute according to OPTION:

       --help display this help and exit

       --version
              output version information and exit

       FORMAT controls the output as in C printf.  Interpreted sequences are:

       \"     double quote

       \\     backslash

       \a     alert (BEL)

       \b     backspace

       \c     produce no further output

       \e     escape

       \f     form feed

       \n     new line

       \r     carriage return

       \t     horizontal tab

       \v     vertical tab

       \NNN   byte with octal value NNN (1 to 3 digits)

       \xHH   byte with hexadecimal value HH (1 to 2 digits)
 Manual page printf(1) line 1/79 39% (press h for help or q to quit)
```

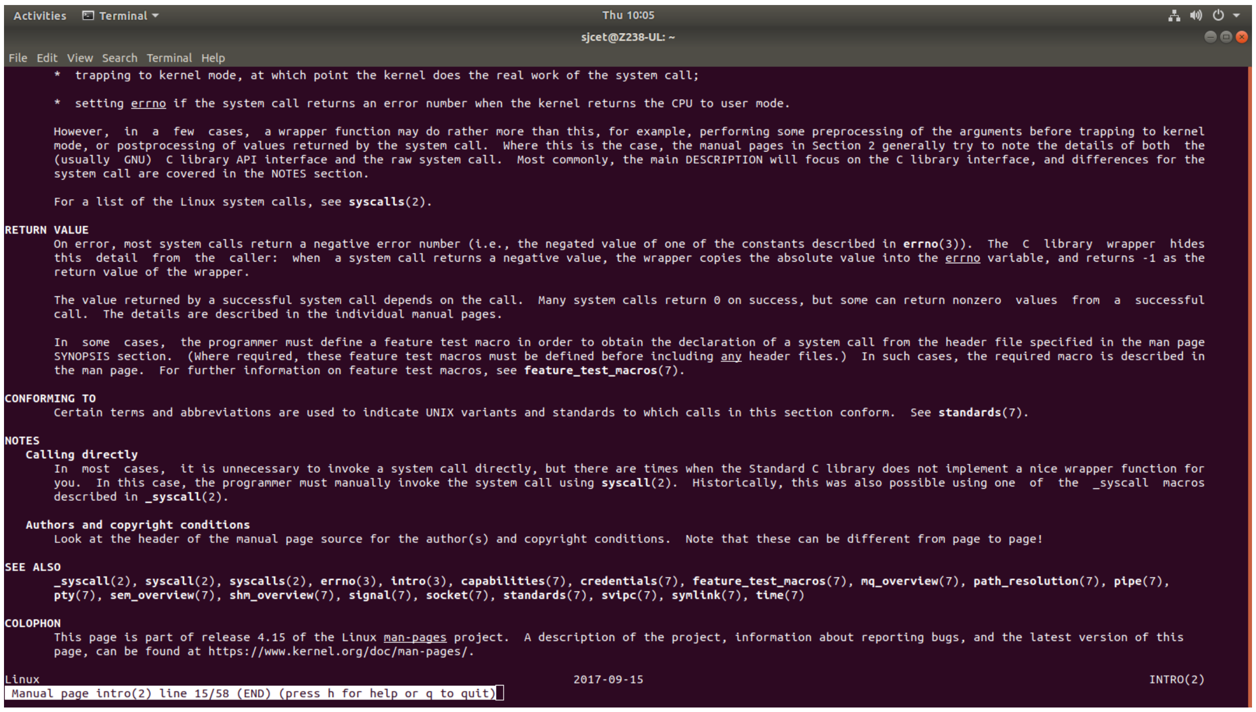


```
       \r     carriage return

       \t     horizontal tab

       \v     vertical tab

       \NNN   byte with octal value NNN (1 to 3 digits)

       \xHH   byte with hexadecimal value HH (1 to 2 digits)

       \uHHHH Unicode (ISO/IEC 10646) character with hex value HHHH (4 digits)

       \UHHHHHHHH
              Unicode character with hex value HHHHHHHH (8 digits)

       %%     a single %

       %b     ARGUMENT as a string with '\' escapes interpreted, except that octal escapes are of the form \0 or \0NNN

       %q     ARGUMENT is printed in a format that can be reused as shell input, escaping non-printable characters with the proposed POSIX $'' syntax.

       and all C format specifications ending with one of diouxXfeEgGcs, with ARGUMENTs converted to proper type first.  Variable widths are handled.

       NOTE:  your  shell  may have its own version of printf, which usually supersedes the version described here.  Please refer to your shell's documentation for details
       about the options it supports.

AUTHOR
       Written by David MacKenzie.

REPORTING BUGS
       GNU coreutils online help: <http://www.gnu.org/software/coreutils/>
       Report printf translation bugs to <http://translationproject.org/team/>

COPYRIGHT
       Copyright © 2017 Free Software Foundation, Inc.  License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>.
       This is free software: you are free to change and redistribute it.  There is NO WARRANTY, to the extent permitted by law.

SEE ALSO
       printf(3)

       Full documentation at: <http://www.gnu.org/software/coreutils/printf>
       or available locally via: info '(coreutils) printf invocation'

GNU coreutils 8.28                              January 2018                                                                PRINTF(1)
 Manual page printf(1) line 36/79 (END) (press h for help or q to quit)
```

cmd: `$ man 2 intro`

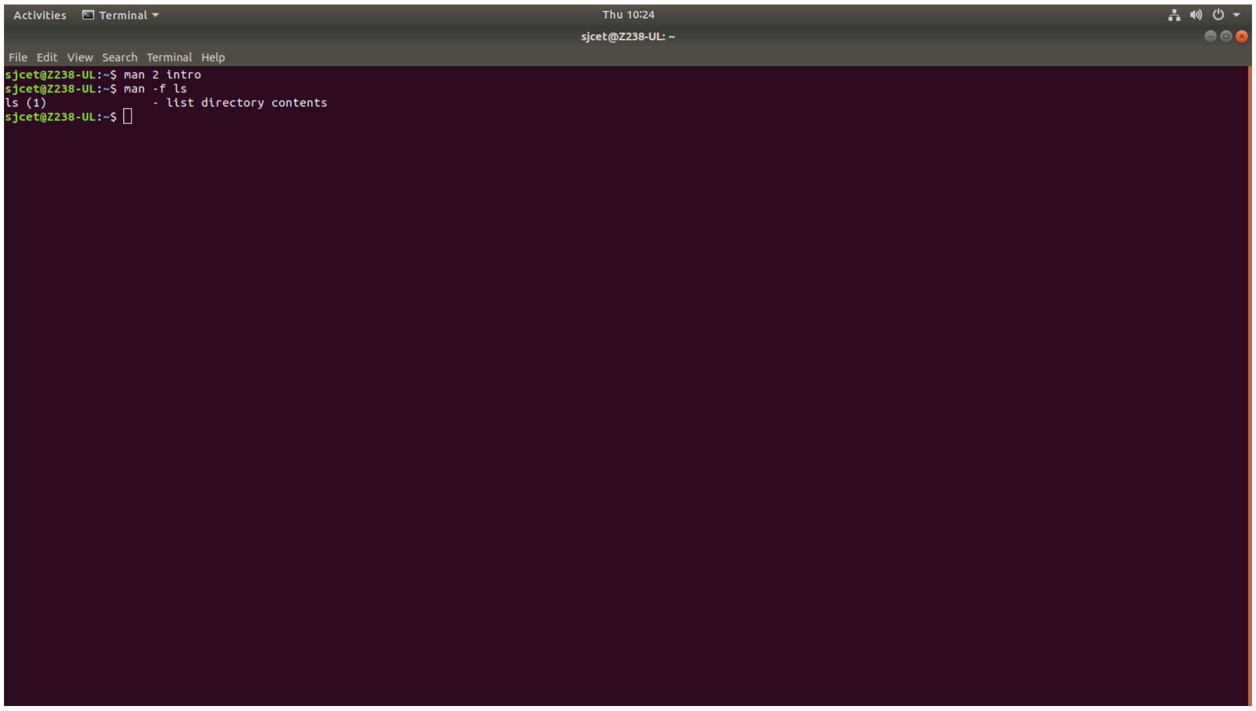


```
INTRO(2)                                         Linux Programmer's Manual                                                    INTRO(2)

NAME
       intro - introduction to system calls

DESCRIPTION
       Section  2  of the manual describes the Linux system calls.  A system call is an entry point into the Linux kernel.  Usually, system calls are not invoked directly:
       instead, most system calls have corresponding C library wrapper functions which perform the steps required (e.g., trapping to kernel mode) in order  to  invoke  the
       system call.  Thus, making a system call looks the same as invoking a normal library function.

       In many cases, the C library wrapper function does nothing more than:

       *  copying arguments and the unique system call number to the registers where the kernel expects them;

       *  trapping to kernel mode, at which point the kernel does the real work of the system call;

       *  setting errno if the system call returns an error number when the kernel returns the CPU to user mode.

       However,  in  a  few  cases,  a wrapper function may do rather more than this, for example, performing some preprocessing of the arguments before trapping to kernel
       mode, or postprocessing of values returned by the system call.  Where this is the case, the manual pages in Section 2 generally try to note the details of both  the
       (usually  GNU)  C library API interface and the raw system call.  Most commonly, the main DESCRIPTION will focus on the C library interface, and differences for the
       system call are covered in the NOTES section.

       For a list of the Linux system calls, see syscalls(2).

RETURN VALUE
       On error, most system calls return a negative error number (i.e., the negated value of one of the constants described in errno(3)).  The  C  library  wrapper  hides
       this  detail  from  the  caller:  when  a system call returns a negative value, the wrapper copies the absolute value into the errno variable, and returns -1 as the
       return value of the wrapper.

       The value returned by a successful system call depends on the call.  Many system calls return 0 on success, but some can return nonzero  values  from  a  successful
       call.  The details are described in the individual manual pages.

       In  some  cases,  the programmer must define a feature test macro in order to obtain the declaration of a system call from the header file specified in the man page
       SYNOPSIS section.  (Where required, these feature test macros must be defined before including any header files.)  In such cases, the required macro is described in
       the man page.  For further information on feature test macros, see feature_test_macros(7).

CONFORMING TO
       Certain terms and abbreviations are used to indicate UNIX variants and standards to which calls in this section conform.  See standards(7).

NOTES
   Calling directly
       In  most  cases,  it is unnecessary to invoke a system call directly, but there are times when the Standard C library does not implement a nice wrapper function for
       you.  In this case, the programmer must manually invoke the system call using syscall(2).  Historically, this was also possible using one  of  the  _syscall  macros
 Manual page intro(2) line 1 (press h for help or q to quit)
```



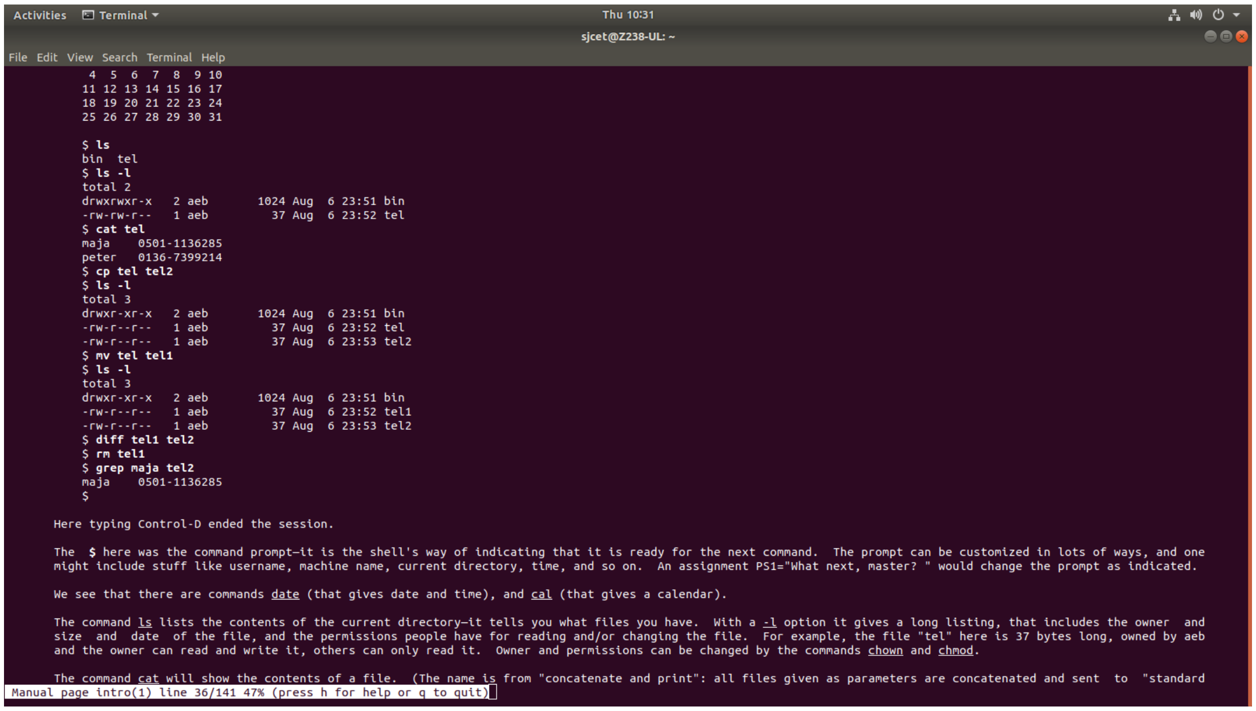
```
       *  trapping to kernel mode, at which point the kernel does the real work of the system call;

       *  setting errno if the system call returns an error number when the kernel returns the CPU to user mode.

       However,  in  a  few  cases,  a wrapper function may do rather more than this, for example, performing some preprocessing of the arguments before trapping to kernel
       mode, or postprocessing of values returned by the system call.  Where this is the case, the manual pages in Section 2 generally try to note the details of both  the
       (usually  GNU)  C library API interface and the raw system call.  Most commonly, the main DESCRIPTION will focus on the C library interface, and differences for the
       system call are covered in the NOTES section.

       For a list of the Linux system calls, see syscalls(2).

RETURN VALUE
       On error, most system calls return a negative error number (i.e., the negated value of one of the constants described in errno(3)).  The  C  library  wrapper  hides
       this  detail  from  the  caller:  when  a system call returns a negative value, the wrapper copies the absolute value into the errno variable, and returns -1 as the
       return value of the wrapper.

       The value returned by a successful system call depends on the call.  Many system calls return 0 on success, but some can return nonzero  values  from  a  successful
       call.  The details are described in the individual manual pages.

       In  some  cases,  the programmer must define a feature test macro in order to obtain the declaration of a system call from the header file specified in the man page
       SYNOPSIS section.  (Where required, these feature test macros must be defined before including any header files.)  In such cases, the required macro is described in
       the man page.  For further information on feature test macros, see feature_test_macros(7).

CONFORMING TO
       Certain terms and abbreviations are used to indicate UNIX variants and standards to which calls in this section conform.  See standards(7).

NOTES
   Calling directly
       In  most  cases,  it is unnecessary to invoke a system call directly, but there are times when the Standard C library does not implement a nice wrapper function for
       you.  In this case, the programmer must manually invoke the system call using syscall(2).  Historically, this was also possible using one  of  the  _syscall  macros
       described in _syscall(2).

   Authors and copyright conditions
       Look at the header of the manual page source for the author(s) and copyright conditions.  Note that these can be different from page to page!

SEE ALSO
       _syscall(2), syscall(2), syscalls(2), errno(3), intro(3), capabilities(7), credentials(7), feature_test_macros(7), mq_overview(7), path_resolution(7), pipe(7),
       pty(7), sem_overview(7), shm_overview(7), signal(7), socket(7), standards(7), svipc(7), symlink(7), time(7)

COLOPHON
       This page is part of release 4.15 of the Linux man-pages project.  A description of the project, information about reporting bugs, and the latest version of this
       page, can be found at https://www.kernel.org/doc/man-pages/.

Linux                                                                     2017-09-15                                                                     INTRO(2)
 Manual page intro(2) line 15/58 (END) (press h for help or q to quit)
```

cmd: $ man -f ls

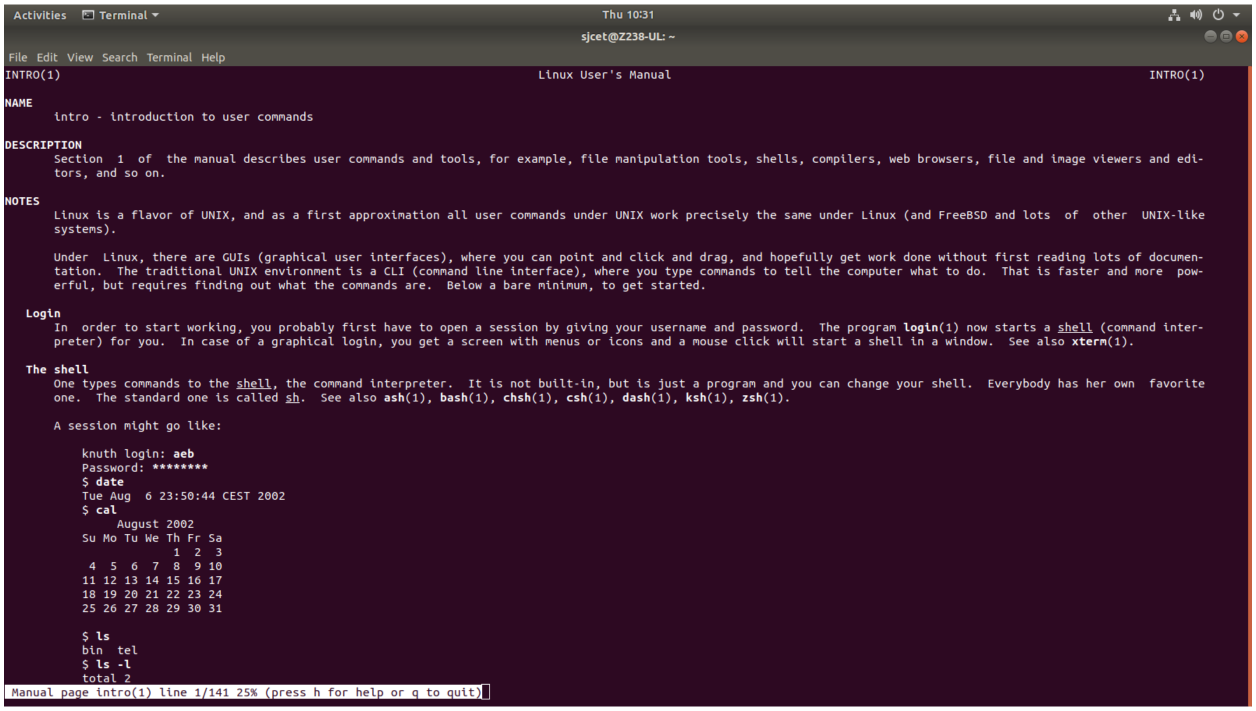


```
sjcet@Z238-UL:~$ man 2 intro
sjcet@Z238-UL:~$ man -f ls
ls (1)               - list directory contents
sjcet@Z238-UL:~$ 
```

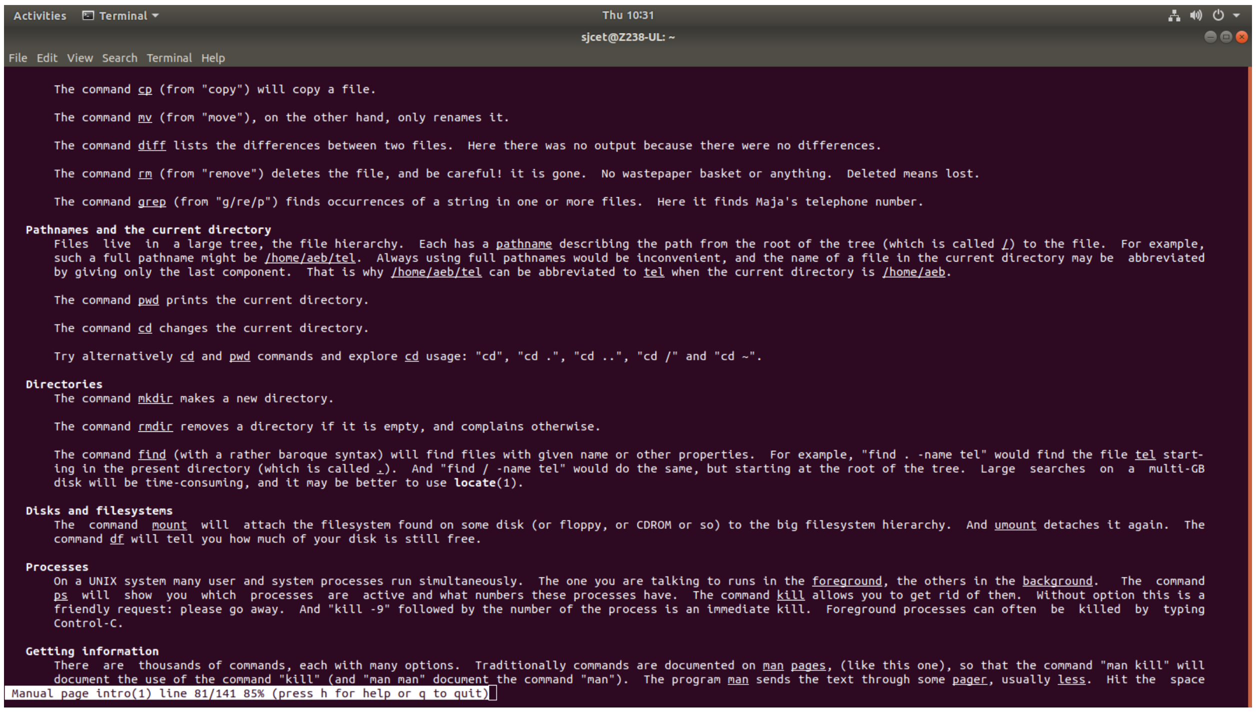
cmd: `$ man -a intro`



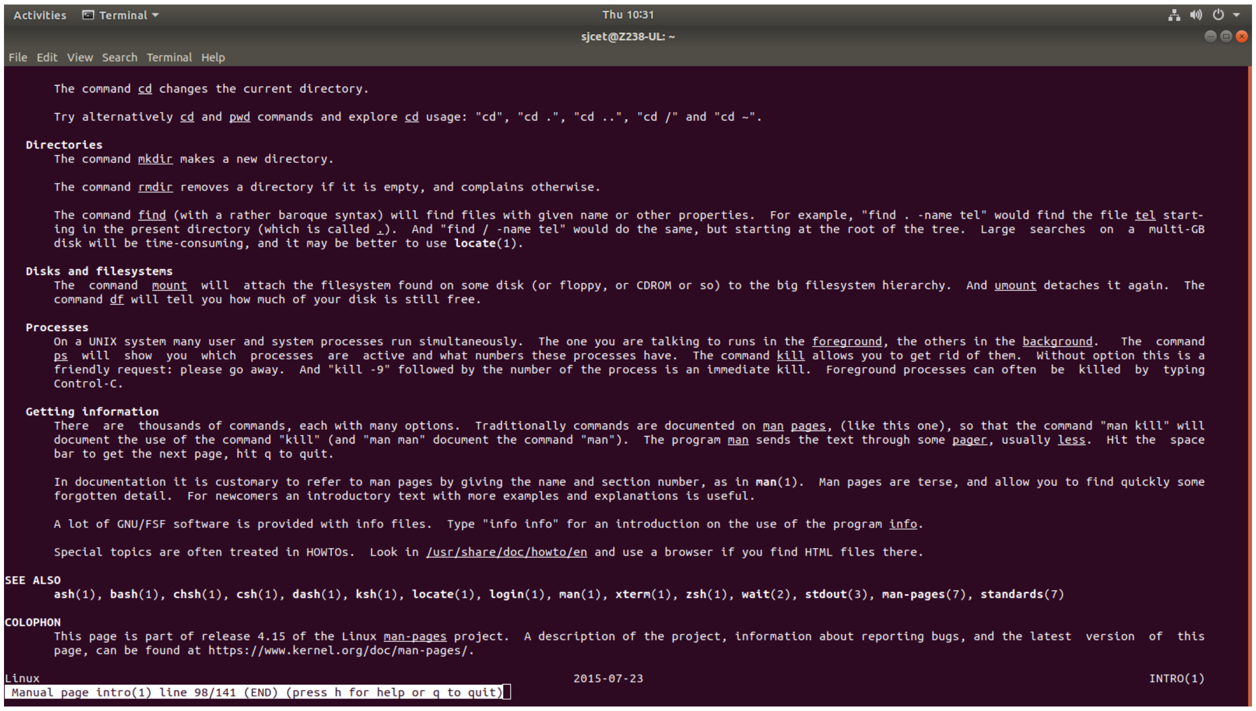
```
          4  5  6  7  8  9 10
         11 12 13 14 15 16 17
         18 19 20 21 22 23 24
         25 26 27 28 29 30 31

         $ ls
         bin  tel
         $ ls -l
         total 2
         drwxrwxr-x   2 aeb       1024 Aug  6 23:51 bin
         -rw-rw-r--   1 aeb         37 Aug  6 23:52 tel
         $ cat tel
         maja    0501-1136285
         peter   0136-7399214
         $ cp tel tel2
         $ ls -l
         total 3
         drwxr-xr-x   2 aeb       1024 Aug  6 23:51 bin
         -rw-r--r--   1 aeb         37 Aug  6 23:52 tel
         -rw-r--r--   1 aeb         37 Aug  6 23:53 tel2
         $ mv tel tel1
         $ ls -l
         total 3
         drwxr-xr-x   2 aeb       1024 Aug  6 23:51 bin
         -rw-r--r--   1 aeb         37 Aug  6 23:52 tel1
         -rw-r--r--   1 aeb         37 Aug  6 23:53 tel2
         $ diff tel1 tel2
         $ rm tel1
         $ grep maja tel2
         maja    0501-1136285
         $

       Here typing Control-D ended the session.

       The  $ here was the command prompt—it is the shell's way of indicating that it is ready for the next command.  The prompt can be customized in lots of ways, and one
       might include stuff like username, machine name, current directory, time, and so on.  An assignment PS1="What next, master? " would change the prompt as indicated.

       We see that there are commands date (that gives date and time), and cal (that gives a calendar).

       The command ls lists the contents of the current directory—it tells you what files you have.  With a -l option it gives a long listing, that includes the owner  and
       size  and  date  of the file, and the permissions people have for reading and/or changing the file.  For example, the file "tel" here is 37 bytes long, owned by aeb
       and the owner can read and write it, others can only read it.  Owner and permissions can be changed by the commands chown and chmod.

       The command cat will show the contents of a file.  (The name is from "concatenate and print": all files given as parameters are concatenated and sent  to  "standard
 Manual page intro(1) line 36/141 47% (press h for help or q to quit)
```



```
INTRO(1)                                                  Linux User's Manual                                                  INTRO(1)

NAME
       intro - introduction to user commands

DESCRIPTION
       Section  1  of  the manual describes user commands and tools, for example, file manipulation tools, shells, compilers, web browsers, file and image viewers and edi-
       tors, and so on.

NOTES
       Linux is a flavor of UNIX, and as a first approximation all user commands under UNIX work precisely the same under Linux (and FreeBSD and lots  of  other  UNIX-like
       systems).

       Under  Linux, there are GUIs (graphical user interfaces), where you can point and click and drag, and hopefully get work done without first reading lots of documen-
       tation.  The traditional UNIX environment is a CLI (command line interface), where you type commands to tell the computer what to do.  That is faster and more  pow-
       erful, but requires finding out what the commands are.  Below a bare minimum, to get started.

   Login
       In  order to start working, you probably first have to open a session by giving your username and password.  The program login(1) now starts a shell (command inter-
       preter) for you.  In case of a graphical login, you get a screen with menus or icons and a mouse click will start a shell in a window.  See also xterm(1).

   The shell
       One types commands to the shell, the command interpreter.  It is not built-in, but is just a program and you can change your shell.  Everybody has her own  favorite
       one.  The standard one is called sh.  See also ash(1), bash(1), chsh(1), csh(1), dash(1), ksh(1), zsh(1).

       A session might go like:

         knuth login: aeb
         Password: ********
         $ date
         Tue Aug  6 23:50:44 CEST 2002
         $ cal
              August 2002
         Su Mo Tu We Th Fr Sa
                      1  2  3
          4  5  6  7  8  9 10
         11 12 13 14 15 16 17
         18 19 20 21 22 23 24
         25 26 27 28 29 30 31

         $ ls
         bin  tel
         $ ls -l
         total 2
 Manual page intro(1) line 1/141 25% (press h for help or q to quit)
```

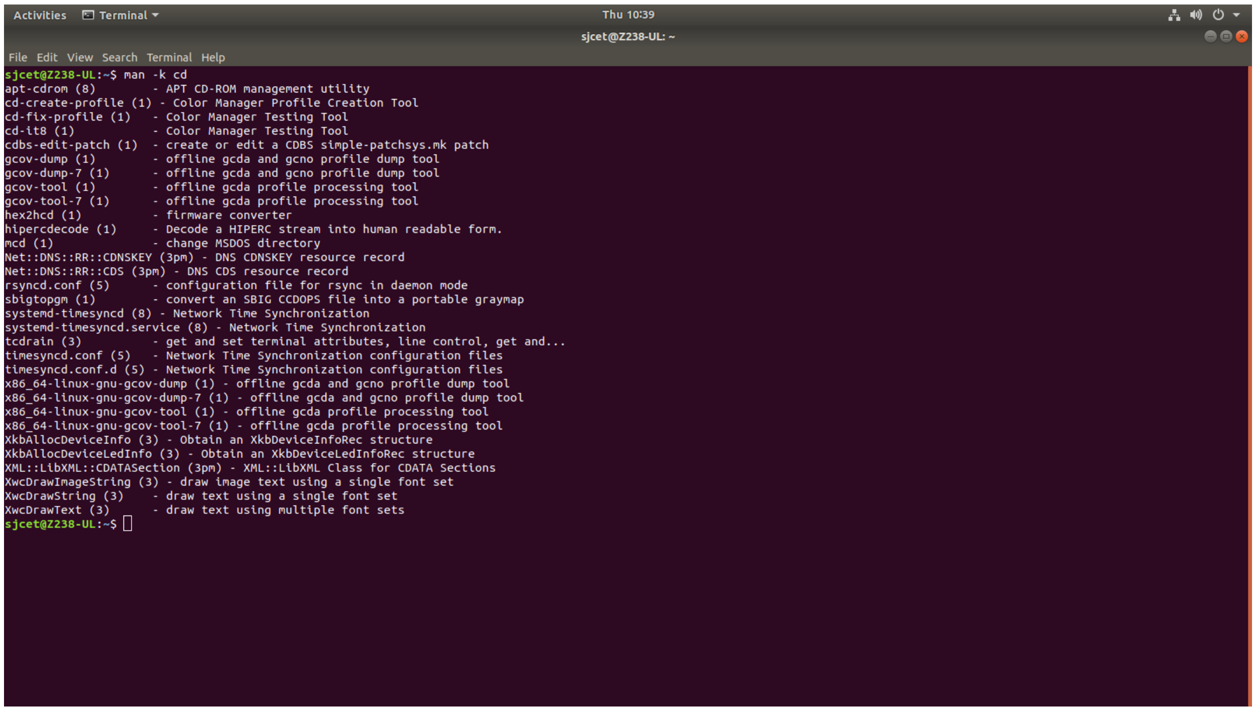


```
       The command cp (from "copy") will copy a file.

       The command mv (from "move"), on the other hand, only renames it.

       The command diff lists the differences between two files.  Here there was no output because there were no differences.

       The command rm (from "remove") deletes the file, and be careful! it is gone.  No wastepaper basket or anything.  Deleted means lost.

       The command grep (from "g/re/p") finds occurrences of a string in one or more files.  Here it finds Maja's telephone number.

   Pathnames and the current directory
       Files  live  in  a large tree, the file hierarchy.  Each has a pathname describing the path from the root of the tree (which is called /) to the file.  For example,
       such a full pathname might be /home/aeb/tel.  Always using full pathnames would be inconvenient, and the name of a file in the current directory may be  abbreviated
       by giving only the last component.  That is why /home/aeb/tel can be abbreviated to tel when the current directory is /home/aeb.

       The command pwd prints the current directory.

       The command cd changes the current directory.

       Try alternatively cd and pwd commands and explore cd usage: "cd", "cd .", "cd ..", "cd /" and "cd ~".

   Directories
       The command mkdir makes a new directory.

       The command rmdir removes a directory if it is empty, and complains otherwise.

       The command find (with a rather baroque syntax) will find files with given name or other properties.  For example, "find . -name tel" would find the file tel start-
       ing in the present directory (which is called .).  And "find / -name tel" would do the same, but starting at the root of the tree.  Large  searches  on  a  multi-GB
       disk will be time-consuming, and it may be better to use locate(1).

   Disks and filesystems
       The  command  mount  will  attach the filesystem found on some disk (or floppy, or CDROM or so) to the big filesystem hierarchy.  And umount detaches it again.  The
       command df will tell you how much of your disk is still free.

   Processes
       On a UNIX system many user and system processes run simultaneously.  The one you are talking to runs in the foreground, the others in the background.   The  command
       ps  will  show  you  which  processes  are  active and what numbers these processes have.  The command kill allows you to get rid of them.  Without option this is a
       friendly request: please go away.  And "kill -9" followed by the number of the process is an immediate kill.  Foreground processes can often  be  killed  by  typing
       Control-C.

   Getting information
       There  are  thousands of commands, each with many options.  Traditionally commands are documented on man pages, (like this one), so that the command "man kill" will
       document the use of the command "kill" (and "man man" document the command "man").  The program man sends the text through some pager, usually less.  Hit the  space
 Manual page intro(1) line 81/141 85% (press h for help or q to quit)
```



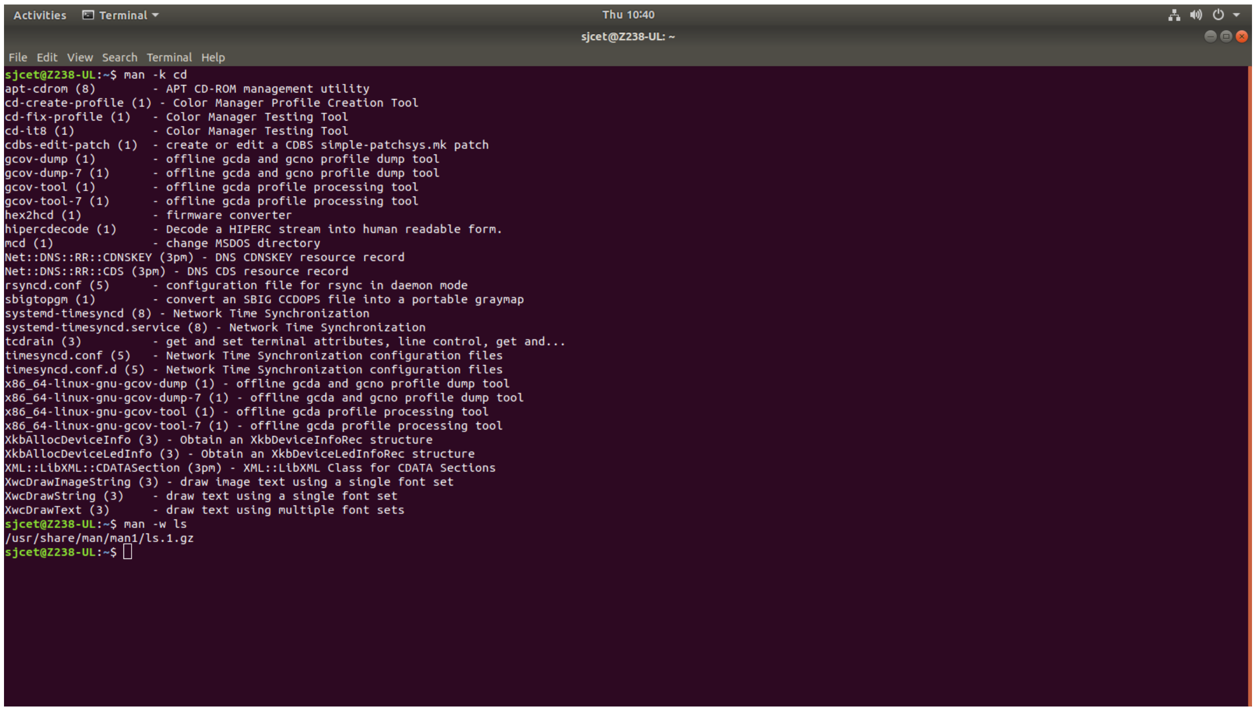
```
       The command cd changes the current directory.

       Try alternatively cd and pwd commands and explore cd usage: "cd", "cd .", "cd ..", "cd /" and "cd ~".

   Directories
       The command mkdir makes a new directory.

       The command rmdir removes a directory if it is empty, and complains otherwise.

       The command find (with a rather baroque syntax) will find files with given name or other properties.  For example, "find . -name tel" would find the file tel start-
       ing in the present directory (which is called .).  And "find / -name tel" would do the same, but starting at the root of the tree.  Large  searches  on  a  multi-GB
       disk will be time-consuming, and it may be better to use locate(1).

   Disks and filesystems
       The  command  mount  will  attach the filesystem found on some disk (or floppy, or CDROM or so) to the big filesystem hierarchy.  And umount detaches it again.  The
       command df will tell you how much of your disk is still free.

   Processes
       On a UNIX system many user and system processes run simultaneously.  The one you are talking to runs in the foreground, the others in the background.   The  command
       ps  will  show  you  which  processes  are  active and what numbers these processes have.  The command kill allows you to get rid of them.  Without option this is a
       friendly request: please go away.  And "kill -9" followed by the number of the process is an immediate kill.  Foreground processes can often  be  killed  by  typing
       Control-C.

   Getting information
       There  are  thousands of commands, each with many options.  Traditionally commands are documented on man pages, (like this one), so that the command "man kill" will
       document the use of the command "kill" (and "man man" document the command "man").  The program man sends the text through some pager, usually less.  Hit the  space
       bar to get the next page, hit q to quit.

       In documentation it is customary to refer to man pages by giving the name and section number, as in man(1).  Man pages are terse, and allow you to find quickly some
       forgotten detail.  For newcomers an introductory text with more examples and explanations is useful.

       A lot of GNU/FSF software is provided with info files.  Type "info info" for an introduction on the use of the program info.

       Special topics are often treated in HOWTOs.  Look in /usr/share/doc/howto/en and use a browser if you find HTML files there.

SEE ALSO
       ash(1), bash(1), chsh(1), csh(1), dash(1), ksh(1), locate(1), login(1), man(1), xterm(1), zsh(1), wait(2), stdout(3), man-pages(7), standards(7)

COLOPHON
       This page is part of release 4.15 of the Linux man-pages project.  A description of the project, information about reporting bugs, and the latest  version  of  this
       page, can be found at https://www.kernel.org/doc/man-pages/.

Linux                                                                 2015-07-23                                                                        INTRO(1)
 Manual page intro(1) line 98/141 (END) (press h for help or q to quit)
```

cmd: $ man -k cd



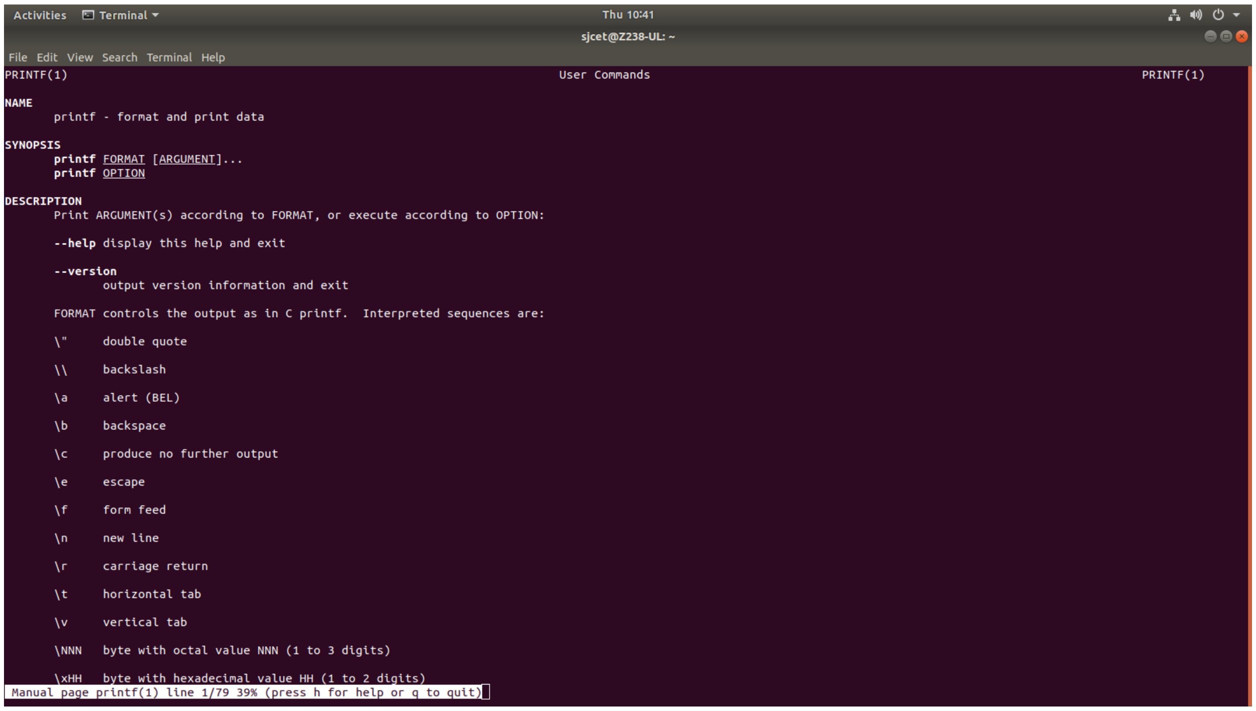
```
sjcet@Z238-UL:~$ man -k cd
apt-cdrom (8)        - APT CD-ROM management utility
cd-create-profile (1) - Color Manager Profile Creation Tool
cd-fix-profile (1)   - Color Manager Testing Tool
cd-it8 (1)           - Color Manager Testing Tool
cdbs-edit-patch (1)  - create or edit a CDBS simple-patchsys.mk patch
gcov-dump (1)        - offline gcda and gcno profile dump tool
gcov-dump-7 (1)      - offline gcda and gcno profile dump tool
gcov-tool (1)        - offline gcda profile processing tool
gcov-tool-7 (1)      - offline gcda profile processing tool
hex2hcd (1)          - firmware converter
hipercdecode (1)     - Decode a HIPERC stream into human readable form.
mcd (1)              - change MSDOS directory
Net::DNS::RR::CDNSKEY (3pm) - DNS CDNSKEY resource record
Net::DNS::RR::CDS (3pm) - DNS CDS resource record
rsyncd.conf (5)      - configuration file for rsync in daemon mode
sbigtopgm (1)        - convert an SBIG CCDOPS file into a portable graymap
systemd-timesyncd (8) - Network Time Synchronization
systemd-timesyncd.service (8) - Network Time Synchronization
tcdrain (3)          - get and set terminal attributes, line control, get and...
timesyncd.conf (5)   - Network Time Synchronization configuration files
timesyncd.conf.d (5) - Network Time Synchronization configuration files
x86_64-linux-gnu-gcov-dump (1) - offline gcda and gcno profile dump tool
x86_64-linux-gnu-gcov-dump-7 (1) - offline gcda and gcno profile dump tool
x86_64-linux-gnu-gcov-tool (1) - offline gcda profile processing tool
x86_64-linux-gnu-gcov-tool-7 (1) - offline gcda profile processing tool
XkbAllocDeviceInfo (3) - Obtain an XkbDeviceInfoRec structure
XkbAllocDeviceLedInfo (3) - Obtain an XkbDeviceLedInfoRec structure
XML::LibXML::CDATASection (3pm) - XML::LibXML Class for CDATA Sections
XwcDrawImageString (3) - draw image text using a single font set
XwcDrawString (3)    - draw text using a single font set
XwcDrawText (3)      - draw text using multiple font sets
sjcet@Z238-UL:~$ 
```

cmd: $ man -w ls

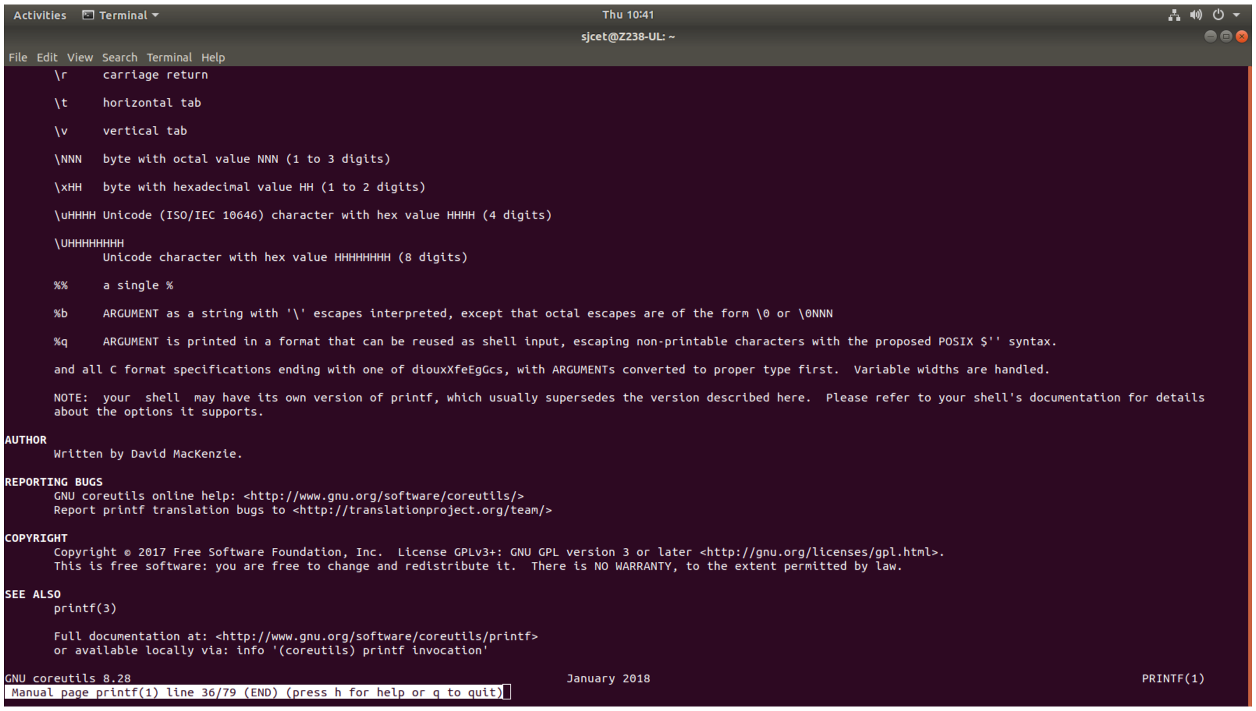


```
sjcet@Z238-UL:~$ man -k cd
apt-cdrom (8)        - APT CD-ROM management utility
cd-create-profile (1) - Color Manager Profile Creation Tool
cd-fix-profile (1)   - Color Manager Testing Tool
cd-it8 (1)           - Color Manager Testing Tool
cdbs-edit-patch (1)  - create or edit a CDBS simple-patchsys.mk patch
gcov-dump (1)        - offline gcda and gcno profile dump tool
gcov-dump-7 (1)      - offline gcda and gcno profile dump tool
gcov-tool (1)        - offline gcda profile processing tool
gcov-tool-7 (1)      - offline gcda profile processing tool
hex2hcd (1)          - firmware converter
hipercdecode (1)     - Decode a HIPERC stream into human readable form.
mcd (1)              - change MSDOS directory
Net::DNS::RR::CDNSKEY (3pm) - DNS CDNSKEY resource record
Net::DNS::RR::CDS (3pm) - DNS CDS resource record
rsyncd.conf (5)      - configuration file for rsync in daemon mode
sbigtopgm (1)        - convert an SBIG CCDOPS file into a portable graymap
systemd-timesyncd (8) - Network Time Synchronization
systemd-timesyncd.service (8) - Network Time Synchronization
tcdrain (3)          - get and set terminal attributes, line control, get and...
timesyncd.conf (5)   - Network Time Synchronization configuration files
timesyncd.conf.d (5) - Network Time Synchronization configuration files
x86_64-linux-gnu-gcov-dump (1) - offline gcda and gcno profile dump tool
x86_64-linux-gnu-gcov-dump-7 (1) - offline gcda and gcno profile dump tool
x86_64-linux-gnu-gcov-tool (1) - offline gcda profile processing tool
x86_64-linux-gnu-gcov-tool-7 (1) - offline gcda profile processing tool
XkbAllocDeviceInfo (3) - Obtain an XkbDeviceInfoRec structure
XkbAllocDeviceLedInfo (3) - Obtain an XkbDeviceLedInfoRec structure
XML::LibXML::CDATASection (3pm) - XML::LibXML Class for CDATA Sections
XwcDrawImageString (3) - draw image text using a single font set
XwcDrawString (3)    - draw text using a single font set
XwcDrawText (3)      - draw text using multiple font sets
sjcet@Z238-UL:~$ man -w ls
/usr/share/man/man1/ls.1.gz
sjcet@Z238-UL:~$ 
```

cmd: $ man -I printf

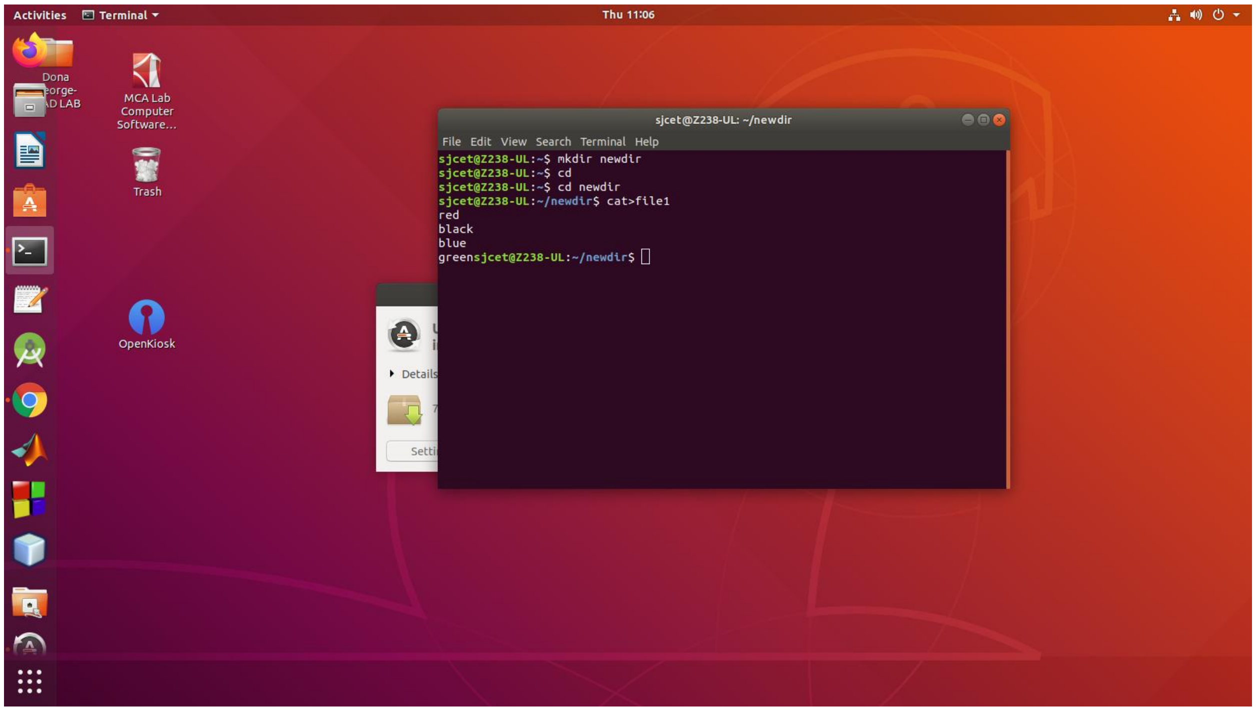


```
PRINTF(1)                                  User Commands                                  PRINTF(1)

NAME
       printf - format and print data

SYNOPSIS
       printf FORMAT [ARGUMENT]...
       printf OPTION

DESCRIPTION
       Print ARGUMENT(s) according to FORMAT, or execute according to OPTION:

       --help display this help and exit

       --version
              output version information and exit

       FORMAT controls the output as in C printf.  Interpreted sequences are:

       \"     double quote

       \\     backslash

       \a     alert (BEL)

       \b     backspace

       \c     produce no further output

       \e     escape

       \f     form feed

       \n     new line

       \r     carriage return

       \t     horizontal tab

       \v     vertical tab

       \NNN   byte with octal value NNN (1 to 3 digits)

       \xHH   byte with hexadecimal value HH (1 to 2 digits)
 Manual page printf(1) line 1/79 39% (press h for help or q to quit)
```



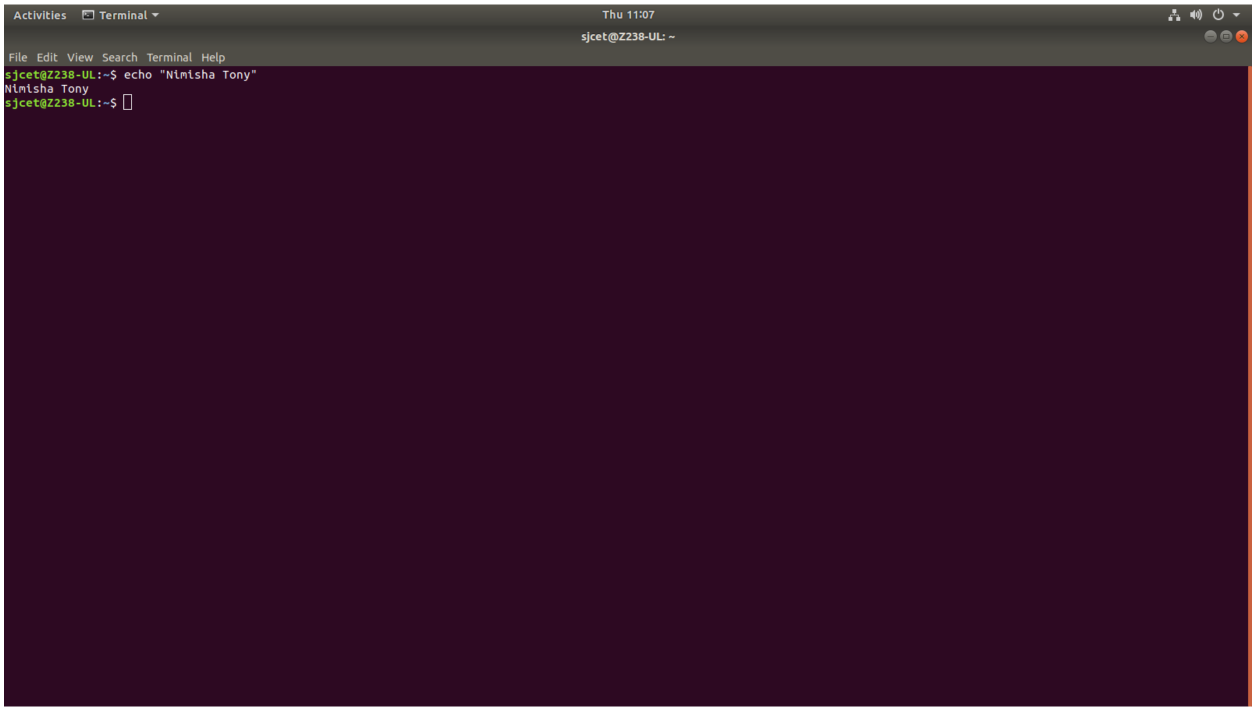
```
       \r     carriage return

       \t     horizontal tab

       \v     vertical tab

       \NNN   byte with octal value NNN (1 to 3 digits)

       \xHH   byte with hexadecimal value HH (1 to 2 digits)

       \uHHHH Unicode (ISO/IEC 10646) character with hex value HHHH (4 digits)

       \UHHHHHHHH
              Unicode character with hex value HHHHHHHH (8 digits)

       %%     a single %

       %b     ARGUMENT as a string with '\' escapes interpreted, except that octal escapes are of the form \0 or \0NNN

       %q     ARGUMENT is printed in a format that can be reused as shell input, escaping non-printable characters with the proposed POSIX $'' syntax.

       and all C format specifications ending with one of diouxXfeEgGcs, with ARGUMENTs converted to proper type first.  Variable widths are handled.

       NOTE:  your  shell  may have its own version of printf, which usually supersedes the version described here.  Please refer to your shell's documentation for details
       about the options it supports.

AUTHOR
       Written by David MacKenzie.

REPORTING BUGS
       GNU coreutils online help: <http://www.gnu.org/software/coreutils/>
       Report printf translation bugs to <http://translationproject.org/team/>

COPYRIGHT
       Copyright © 2017 Free Software Foundation, Inc.  License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>.
       This is free software: you are free to change and redistribute it.  There is NO WARRANTY, to the extent permitted by law.

SEE ALSO
       printf(3)

       Full documentation at: <http://www.gnu.org/software/coreutils/printf>
       or available locally via: info '(coreutils) printf invocation'

GNU coreutils 8.28                         January 2018                                   PRINTF(1)
 Manual page printf(1) line 36/79 (END) (press h for help or q to quit)
```

2_cmd:
mkdir newdir
Cd newdir
cat>files



```
sjcet@Z238-UL:~$ mkdir newdir
sjcet@Z238-UL:~$ cd
sjcet@Z238-UL:~$ cd newdir
sjcet@Z238-UL:~/newdir$ cat>file1
red
black
blue
greensjcet@Z238-UL:~/newdir$
```

Cmd: echo



```
sjcet@Z238-UL:~$ echo "Nimisha Tony"
Nimisha Tony
sjcet@Z238-UL:~$
```