**OS ASSIGNMENT 1A**

**Name:Mansi Mokashi**

**Roll no:87**

**TE IT**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Basic Commands:**

~$ echo $ SHELL

$ SHELL

~$ date --help

Usage: date [OPTION]... [+FORMAT] or: date [-u|--utc|--universal] [MMDDhhmm[[CC]YY][.ss]]

Display the current time in the given FORMAT, or set the system date.

Mandatory arguments to long options are mandatory for short options too.

-d, --date=STRING display time described by STRING, not 'now'

--debug annotate the parsed date, and warn about questionable usage to stderr -f, --file=DATEFILE like --date; once for each line of DATEFILE -I[FMT], --iso-8601[=FMT] output date/time in ISO 8601 format.

FMT='date' for date only (the default), 'hours', 'minutes', 'seconds', or 'ns' for date and time to the indicated precision. Example: 2006-08-14T02:34:56-06:00 -R, --rfc-email output date and time in RFC 5322 format. Example: Mon, 14 Aug 2006 02:34:56 -0600 --rfc-3339=FMT output date/time in RFC 3339 format.

FMT='date', 'seconds', or 'ns' for date and time to the indicated precision. Example: 2006-08-14 02:34:56-06:00

-r, --reference=FILE display the last modification time of FILE

-s, --set=STRING set time described by STRING

-u, --utc, --universal print or set Coordinated Universal Time (UTC)

--help display this help and exit

--version output version information and exit

FORMAT controls the output. Interpreted sequences are:

%% a literal %

%a locale's abbreviated weekday name (e.g., Sun)

%A locale's full weekday name (e.g., Sunday)

%b locale's abbreviated month name (e.g., Jan)

%B locale's full month name (e.g., January)

%c locale's date and time (e.g., Thu Mar 3 23:05:25 2005)

%C century; like %Y, except omit last two digits (e.g., 20)

%d day of month (e.g., 01)

%D date; same as %m/%d/%y

%e day of month, space padded; same as %\_d

%F full date; same as %Y-%m-%d

%g last two digits of year of ISO week number (see %G)

%G year of ISO week number (see %V); normally useful only with %V

%h same as %b

%H hour (00..23)

%I hour (01..12)

%j day of year (001..366)

%k hour, space padded ( 0..23); same as %\_H

%l hour, space padded ( 1..12); same as %\_I

%m month (01..12)

%M minute (00..59)

%n a newline

%N nanoseconds (000000000..999999999)

%p locale's equivalent of either AM or PM; blank if not known

%P like %p, but lower case

%q quarter of year (1..4)

%r locale's 12-hour clock time (e.g., 11:11:04 PM)

%R 24-hour hour and minute; same as %H:%M

%s seconds since 1970-01-01 00:00:00 UTC

%S second (00..60)

%t a tab

%T time; same as %H:%M:%S

%u day of week (1..7); 1 is Monday

%U week number of year, with Sunday as first day of week (00..53)

%V ISO week number, with Monday as first day of week (01..53)

%w day of week (0..6); 0 is Sunday

%W week number of year, with Monday as first day of week (00..53)

%x locale's date representation (e.g., 12/31/99)

%X locale's time representation (e.g., 23:13:48)

%y last two digits of year (00..99)

%Y year

%z +hhmm numeric time zone (e.g., -0400)

%:z +hh:mm numeric time zone (e.g., -04:00)

%::z +hh:mm:ss numeric time zone (e.g., -04:00:00)

%:::z numeric time zone with : to necessary precision (e.g., -04, +05:30)

%Z alphabetic time zone abbreviation (e.g., EDT)

By default, date pads numeric fields with zeroes. The following optional flags may follow '%':

- (hyphen) do not pad the field

\_ (underscore) pad with spaces

0 (zero) pad with zeros

^ use upper case if possible

# use opposite case if possible

After any flags comes an optional field width, as a decimal number; then an optional modifier, which is either

E to use the locale's alternate representations if available, or O to use the locale's alternate numeric symbols if available.

Examples:

Convert seconds since the epoch (1970-01-01 UTC) to a date

$ date --date='@2147483647'

Show the time on the west coast of the US (use tzselect(1) to find TZ)

$ TZ='America/Los\_Angeles' date

Show the local time for 9AM next Friday on the west coast of the US $ date --date='TZ="America/Los\_Angeles" 09:00 next Fri'

GNU coreutils online help: <https://www.gnu.org/software/coreutils/>

Report date translation bugs to <https://translationproject.org/team/> Full documentation at: <https://www.gnu.org/software/coreutils/date> or available locally via: info '(coreutils) date invocation'

~$ date

Tue Jul 20 09:26:42 UTC 2021

~$ who

~$ pwd

/home/user

~$ echo "Today is date"

Today is date

~$ echo "Today is `date`"

Today is Tue oct 13 09:42:22 UTC 2022

~$ echo "expr 6+3'

>

> echo "expr 6+3"

> ~$ echo "expr 6+3"

~$ ls

1. 2022-10-20-145630.term 2021-08-31-143145.term 2022-10-28-154842.x11

Assignment2A.c Fun.c assign2bparent.c fork3.c 'practical3(2).term' '~$'

1. 2021-07-20-145846.term 2021-08-31-143436.x11 2021-10-13-160842.term

Assignment2A.x11 Practical3 childassign2b.c forloop.term practice1.c

2021-07-13-150546.term 2021-07-20-150523.term 2022-09-28-131820.x11

2021-10-13-161353.x11 Assignment2B.x11 Practical3.term echo 'grep command' sec.c

2021-07-13-151039.term 2021-07-20-150753.term 2022-09-28-142218.x11 Assign1srk.c

Assignment2b.c PracticepD.x11 file mtable testfor

2021-07-13-151653.term 2021-07-20-154152.term 2021-09-28-143728 Assign2bchild.c

Assignment2b.x11 a.out first myfile then

2021-07-20-144738.term 2021-07-28-111759 2021-09-28-143825.x11 Assign2bparent.c

Assignment2bos.x11 abc first.c parentassign2b.c third.c

~$ cat>first

#

#MY first Shell Script

#

clear

echo "Knowledge is Power

~$ expr 6 + 3

9

~$ echo $OSTYPE linux-gnu

~$ echo $PATH

/cocalc/bin:/cocalc/src/smc-project/bin:/home/user/bin:/home/user/.local/bin:/ext/bin:/usr/lib/xpra:

/opt/ghc/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/ext/data/homer/bin:/ext/data /weblogo:/usr/lib/postgresql/10/bin

~$ echo $SHELL

/bin/bash

~$ x=Pvg

~$ echo $x Pvg

~$ cat > testfor for i in 1 2 3 4 5

do echo "Welcome $i times" done~$ chmod +x testfor

~$ ./testfor

Welcome 1 times

Welcome 2 times

Welcome 3 times

Welcome 4 times

Welcome 5 times

~$ cat > mtable

#!/bin/sh

#

#Script to test for loop

#

#

if [ $# -eq 0 ] then

echo "Error - Number missing form command line argument" echo "Syntax : $0 number" echo " Use to print multiplication table for given number" exit 1 fi

n=$1

for i in 1 2 3 4 5 6 7 8 9 10

do

echo "$n \* $i = `expr $i \\* $n`" done

bash: e: command not found

~$ chmod +x mtable

~$ ./mtable 3

3 \* 1 = 3

3 \* 2 = 6

3 \* 3 = 9

3 \* 4 = 12

3 \* 5 = 15

3 \* 6 = 18

3 \* 7 = 21

3 \* 8 = 24

3 \* 9 = 27

3 \* 10 = 30

~$ ./mtable 11

11 \* 1 = 11

11 \* 2 = 22

11 \* 3 = 33

11 \* 4 = 44

11 \* 5 = 55

11 \* 6 = 66

11 \* 7 = 77

11 \* 8 = 88

11 \* 9 = 99

11 \* 10 = 110

~$ a=0

~$ while [ "$a" -lt 10 ] # this is loop1

> do

> b="$a"

> while [ "$b" -ge 0 ] # this is loop2

> do

> echo -n "$b "

> b=`expr $b - 1`

> done > echo

> a=`expr $a + 1`

> done

0

1. 0
2. 1 0
3. 2 1 0
4. 3 2 1 0
5. 4 3 2 1 0
6. 5 4 3 2 1 0
7. 6 5 4 3 2 1 0
8. 7 6 5 4 3 2 1 0
9. 8 7 6 5 4 3 2 1 0

~$ grep helllo myfile

~$ grep ^D myfile

~$ cat myfile

^D

~$ cat > abc pvgcoet is my college pune is the cit~$ grep pune abc pune is the city

~$

~$ grep fox myfile

~$ grep fox

first

~$ [ 1 -eq 1 ] && [ -n "`echo true 1>&2`" ] true ~$ if [ "$a" != "$b" ]

> then

> echo "$a is not equal to $b."

> echo "(Program ends)" > .

> fi

5 is not equal to 2. (Program ends) bash: .: filename argument required .: usage: . filename [arguments]

~$ read name;echo "hello $name" Mansi hello Mansi ~$ pwd

/home/user

~$ grep myfile

HELLO THERE