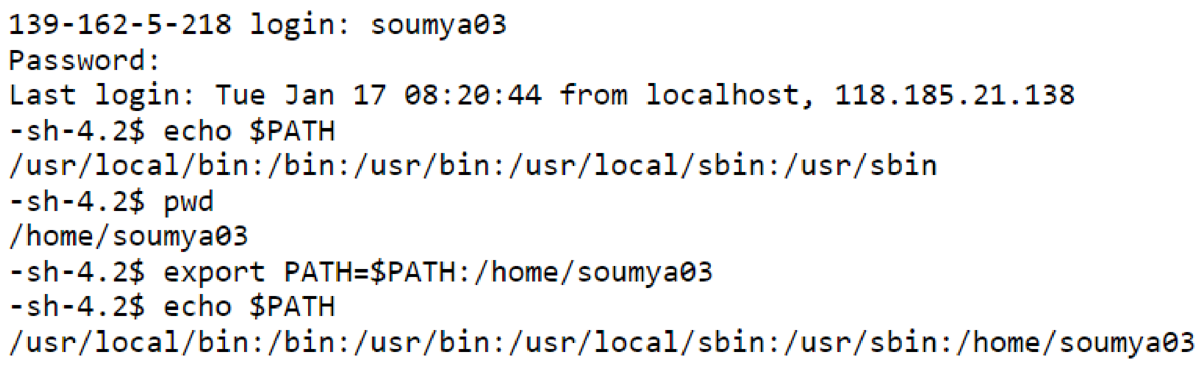
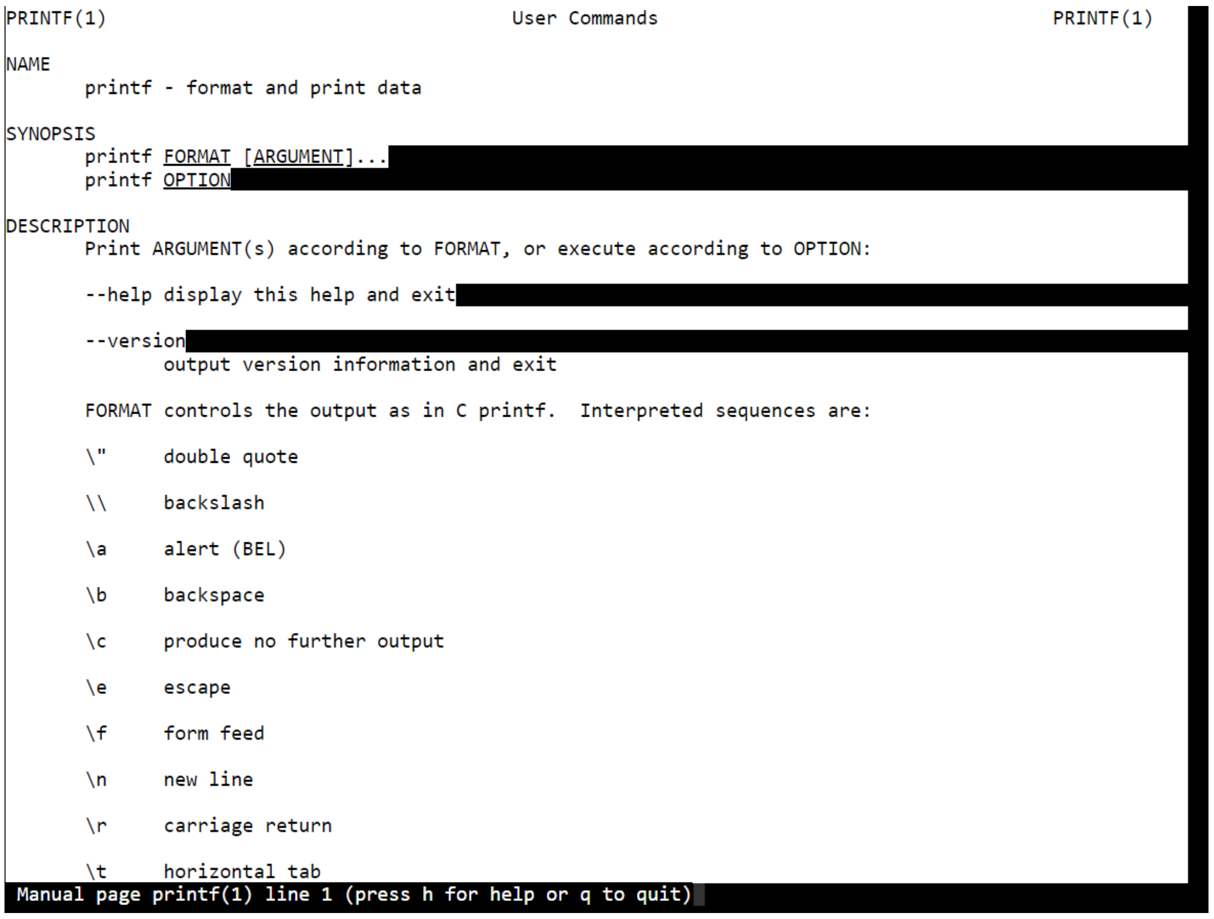
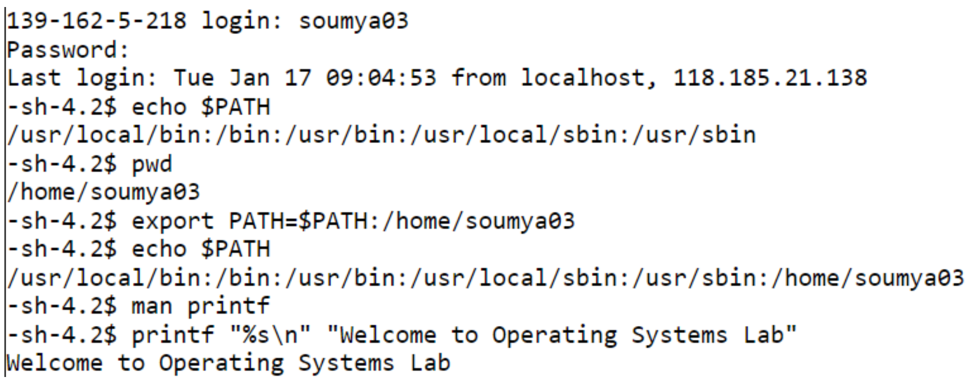


```
139-162-5-218 login: soumya03
Password:
Last login: Tue Jan 17 08:20:44 from localhost, 118.185.21.138
-sh-4.2$ echo $PATH
/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin
-sh-4.2$ pwd
/home/soumya03
-sh-4.2$ export PATH=$PATH:/home/soumya03
-sh-4.2$ echo $PATH
/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/home/soumya03
```



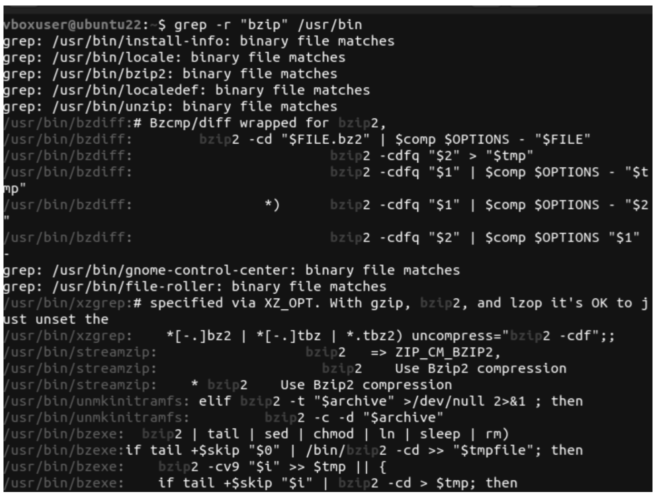
```
PRINTF(1)                        User Commands                       PRINTF(1)

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
 Manual page printf(1) line 1 (press h for help or q to quit)
```



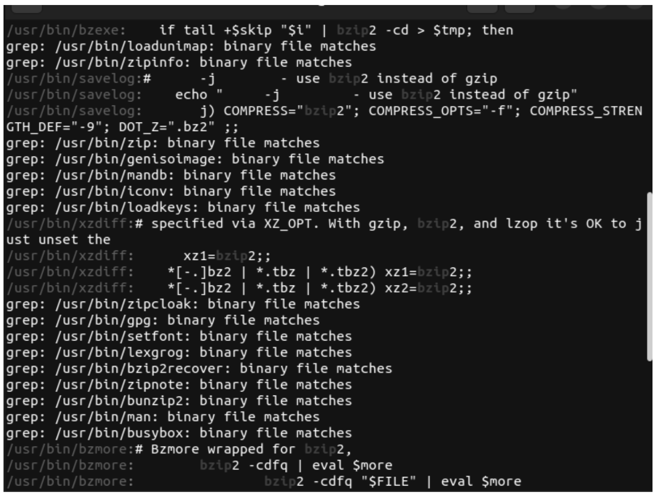
```
139-162-5-218 login: soumya03
Password:
Last login: Tue Jan 17 09:04:53 from localhost, 118.185.21.138
-sh-4.2$ echo $PATH
/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin
-sh-4.2$ pwd
/home/soumya03
-sh-4.2$ export PATH=$PATH:/home/soumya03
-sh-4.2$ echo $PATH
/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/home/soumya03
-sh-4.2$ man printf
-sh-4.2$ printf "%s\n" "Welcome to Operating Systems Lab"
Welcome to Operating Systems Lab
```

E21CSEU0760

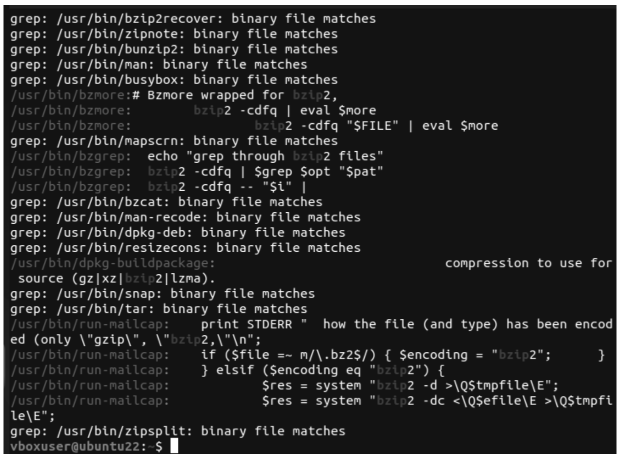
LAB 2



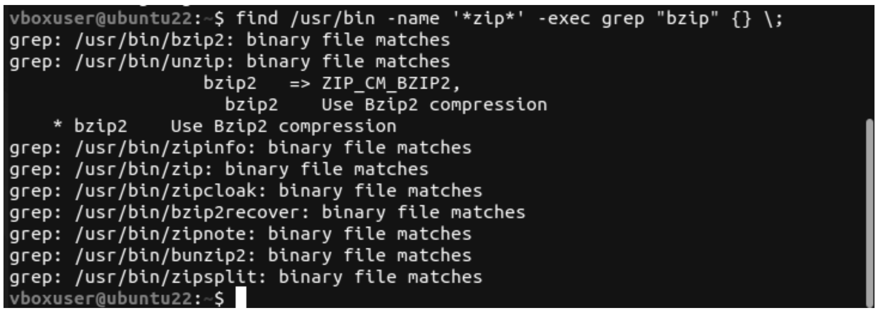
```
vboxuser@ubuntu22:~$ grep -r "bzip" /usr/bin
grep: /usr/bin/install-info: binary file matches
grep: /usr/bin/locale: binary file matches
grep: /usr/bin/bzip2: binary file matches
grep: /usr/bin/localedef: binary file matches
grep: /usr/bin/unzip: binary file matches
/usr/bin/bzdiff:# Bzcmp/diff wrapped for bzip2,
/usr/bin/bzdiff:        bzip2 -cd "$FILE.bz2" | $comp $OPTIONS - "$FILE"
/usr/bin/bzdiff:                        bzip2 -cdfq "$2" > "$tmp"
/usr/bin/bzdiff:                        bzip2 -cdfq "$1" | $comp $OPTIONS - "$t
mp"
/usr/bin/bzdiff:                *)      bzip2 -cdfq "$1" | $comp $OPTIONS - "$2
"
/usr/bin/bzdiff:                        bzip2 -cdfq "$2" | $comp $OPTIONS "$1"
-
grep: /usr/bin/gnome-control-center: binary file matches
grep: /usr/bin/file-roller: binary file matches
/usr/bin/xzgrep:# specified via XZ_OPT. With gzip, bzip2, and lzop it's OK to j
ust unset the
/usr/bin/xzgrep:    *[-.]bz2 | *[-.]tbz | *.tbz2) uncompress="bzip2 -cdf";;
/usr/bin/streamzip:                   bzip2   => ZIP_CM_BZIP2,
/usr/bin/streamzip:                     bzip2    Use Bzip2 compression
/usr/bin/streamzip:    * bzip2    Use Bzip2 compression
/usr/bin/unmkinitramfs: elif bzip2 -t "$archive" >/dev/null 2>&1 ; then
/usr/bin/unmkinitramfs:         bzip2 -c -d "$archive"
/usr/bin/bzexe:  bzip2 | tail | sed | chmod | ln | sleep | rm)
/usr/bin/bzexe:if tail +$skip "$0" | /bin/bzip2 -cd >> "$tmpfile"; then
/usr/bin/bzexe:    bzip2 -cv9 "$i" >> $tmp || {
/usr/bin/bzexe:    if tail +$skip "$i" | bzip2 -cd > $tmp; then
```



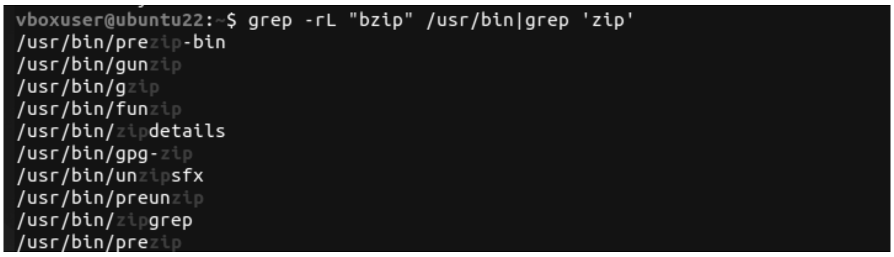
```
/usr/bin/bzexe:    if tail +$skip "$i" | bzip2 -cd > $tmp; then
grep: /usr/bin/loadunimap: binary file matches
grep: /usr/bin/zipinfo: binary file matches
/usr/bin/savelog:#      -j        - use bzip2 instead of gzip
/usr/bin/savelog:    echo "     -j          - use bzip2 instead of gzip"
/usr/bin/savelog:       j) COMPRESS="bzip2"; COMPRESS_OPTS="-f"; COMPRESS_STREN
GTH_DEF="-9"; DOT_Z=".bz2" ;;
grep: /usr/bin/zip: binary file matches
grep: /usr/bin/genisoimage: binary file matches
grep: /usr/bin/mandb: binary file matches
grep: /usr/bin/iconv: binary file matches
grep: /usr/bin/loadkeys: binary file matches
/usr/bin/xzdiff:# specified via XZ_OPT. With gzip, bzip2, and lzop it's OK to j
ust unset the
/usr/bin/xzdiff:      xz1=bzip2;;
/usr/bin/xzdiff:    *[-.]bz2 | *.tbz | *.tbz2) xz1=bzip2;;
/usr/bin/xzdiff:    *[-.]bz2 | *.tbz | *.tbz2) xz2=bzip2;;
grep: /usr/bin/zipcloak: binary file matches
grep: /usr/bin/gpg: binary file matches
grep: /usr/bin/setfont: binary file matches
grep: /usr/bin/lexgrog: binary file matches
grep: /usr/bin/bzip2recover: binary file matches
grep: /usr/bin/zipnote: binary file matches
grep: /usr/bin/bunzip2: binary file matches
grep: /usr/bin/man: binary file matches
grep: /usr/bin/busybox: binary file matches
/usr/bin/bzmore:# Bzmore wrapped for bzip2,
/usr/bin/bzmore:        bzip2 -cdfq | eval $more
/usr/bin/bzmore:                bzip2 -cdfq "$FILE" | eval $more
```



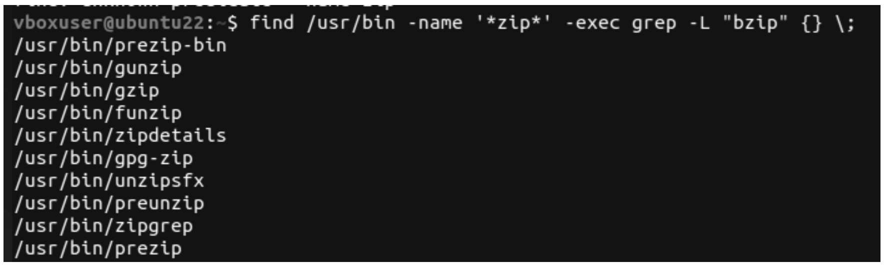
```
grep: /usr/bin/bzip2recover: binary file matches
grep: /usr/bin/zipnote: binary file matches
grep: /usr/bin/bunzip2: binary file matches
grep: /usr/bin/man: binary file matches
grep: /usr/bin/busybox: binary file matches
/usr/bin/bzmore:# Bzmore wrapped for bzip2,
/usr/bin/bzmore:        bzip2 -cdfq | eval $more
/usr/bin/bzmore:                bzip2 -cdfq "$FILE" | eval $more
grep: /usr/bin/mapscrn: binary file matches
/usr/bin/bzgrep:  echo "grep through bzip2 files"
/usr/bin/bzgrep:  bzip2 -cdfq | $grep $opt "$pat"
/usr/bin/bzgrep:  bzip2 -cdfq -- "$i" |
grep: /usr/bin/bzcat: binary file matches
grep: /usr/bin/man-recode: binary file matches
grep: /usr/bin/dpkg-deb: binary file matches
grep: /usr/bin/resizecons: binary file matches
/usr/bin/dpkg-buildpackage:                              compression to use for
 source (gz|xz|bzip2|lzma).
grep: /usr/bin/snap: binary file matches
grep: /usr/bin/tar: binary file matches
/usr/bin/run-mailcap:    print STDERR "  how the file (and type) has been encod
ed (only \"gzip\", \"bzip2,\"\n";
/usr/bin/run-mailcap:    if ($file =~ m/\.bz2$/) { $encoding = "bzip2";      }
/usr/bin/run-mailcap:    } elsif ($encoding eq "bzip2") {
/usr/bin/run-mailcap:            $res = system "bzip2 -d >\Q$tmpfile\E";
/usr/bin/run-mailcap:            $res = system "bzip2 -dc <\Q$efile\E >\Q$tmpfi
le\E";
grep: /usr/bin/zipsplit: binary file matches
vboxuser@ubuntu22:~$
```



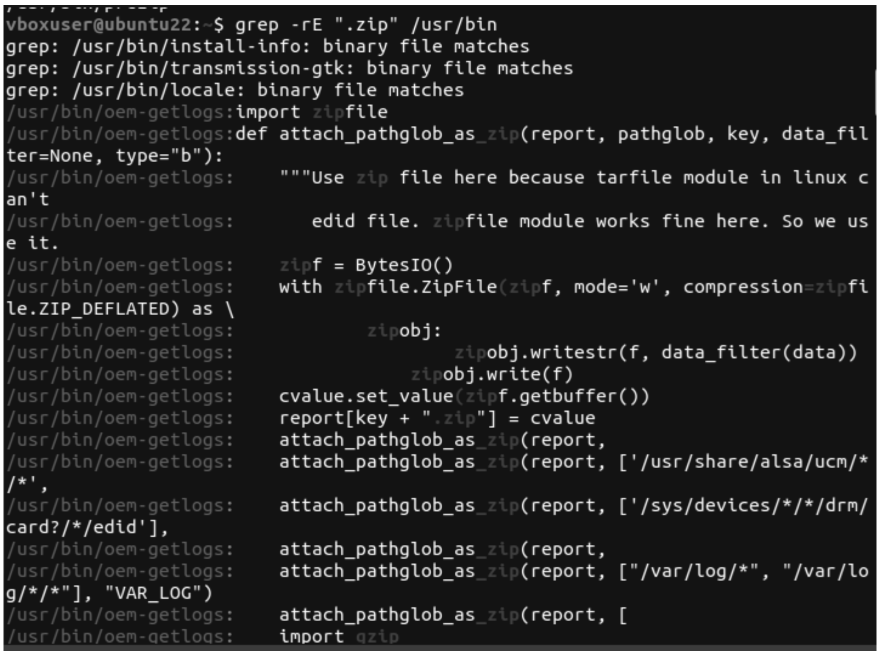
```
vboxuser@ubuntu22:~$ find /usr/bin -name '*zip*' -exec grep "bzip" {} \;
grep: /usr/bin/bzip2: binary file matches
grep: /usr/bin/unzip: binary file matches
                  bzip2   => ZIP_CM_BZIP2,
                    bzip2    Use Bzip2 compression
    * bzip2    Use Bzip2 compression
grep: /usr/bin/zipinfo: binary file matches
grep: /usr/bin/zip: binary file matches
grep: /usr/bin/zipcloak: binary file matches
grep: /usr/bin/bzip2recover: binary file matches
grep: /usr/bin/zipnote: binary file matches
grep: /usr/bin/bunzip2: binary file matches
grep: /usr/bin/zipsplit: binary file matches
vboxuser@ubuntu22:~$
```



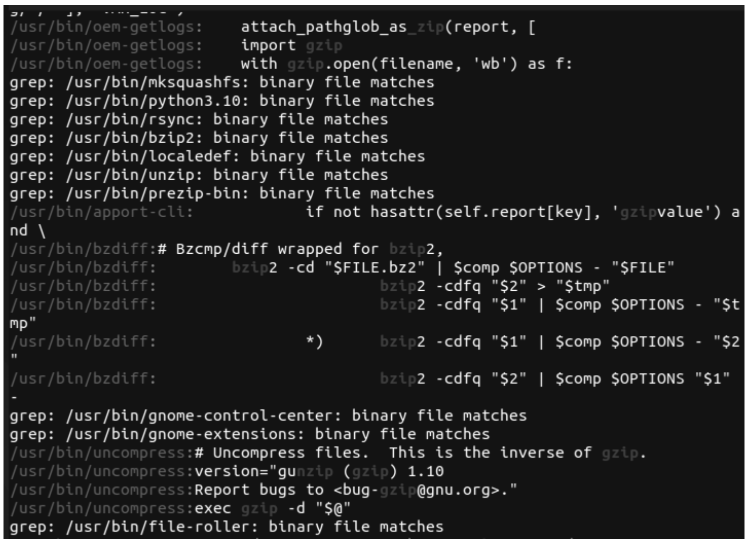
```
vboxuser@ubuntu22:~$ grep -rL "bzip" /usr/bin|grep 'zip'
/usr/bin/prezip-bin
/usr/bin/gunzip
/usr/bin/gzip
/usr/bin/funzip
/usr/bin/zipdetails
/usr/bin/gpg-zip
/usr/bin/unzipsfx
/usr/bin/preunzip
/usr/bin/zipgrep
/usr/bin/prezip
```



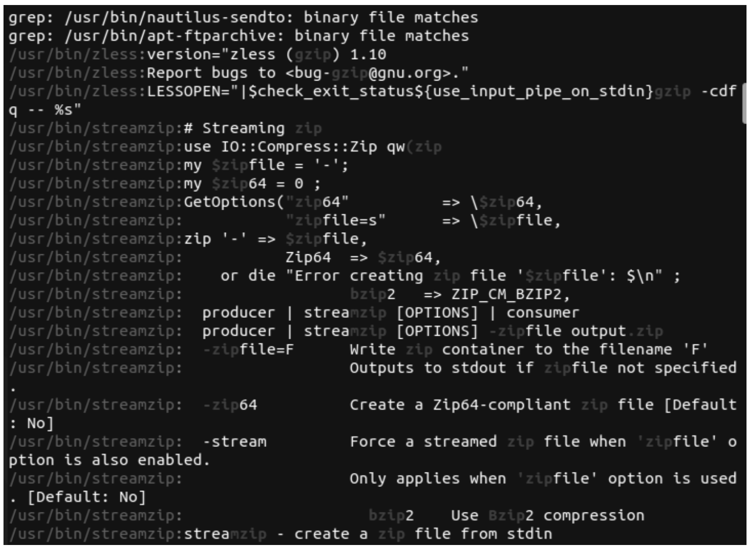
```
vboxuser@ubuntu22:~$ find /usr/bin -name '*zip*' -exec grep -L "bzip" {} \;
/usr/bin/prezip-bin
/usr/bin/gunzip
/usr/bin/gzip
/usr/bin/funzip
/usr/bin/zipdetails
/usr/bin/gpg-zip
/usr/bin/unzipsfx
/usr/bin/preunzip
/usr/bin/zipgrep
/usr/bin/prezip
```



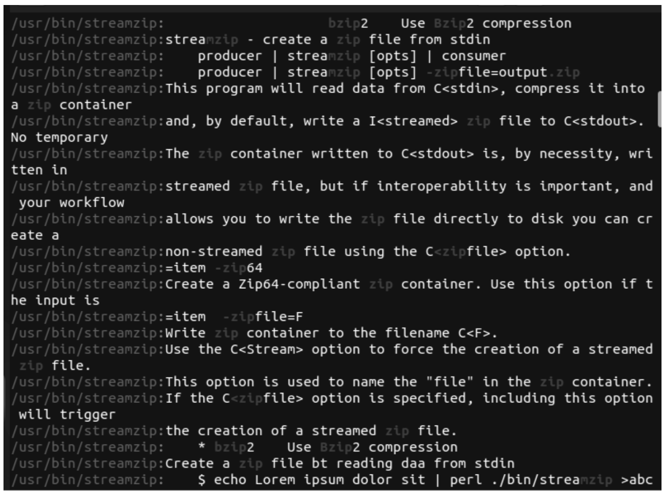
```
vboxuser@ubuntu22:~$ grep -rE ".zip" /usr/bin
grep: /usr/bin/install-info: binary file matches
grep: /usr/bin/transmission-gtk: binary file matches
grep: /usr/bin/locale: binary file matches
/usr/bin/oem-getlogs:import zipfile
/usr/bin/oem-getlogs:def attach_pathglob_as_zip(report, pathglob, key, data_fil
ter=None, type="b"):
/usr/bin/oem-getlogs:    """Use zip file here because tarfile module in linux c
an't
/usr/bin/oem-getlogs:       edid file. zipfile module works fine here. So we us
e it.
/usr/bin/oem-getlogs:    zipf = BytesIO()
/usr/bin/oem-getlogs:    with zipfile.ZipFile(zipf, mode='w', compression=zipfi
le.ZIP_DEFLATED) as \
/usr/bin/oem-getlogs:            zipobj:
/usr/bin/oem-getlogs:                    zipobj.writestr(f, data_filter(data))
/usr/bin/oem-getlogs:                zipobj.write(f)
/usr/bin/oem-getlogs:    cvalue.set_value(zipf.getbuffer())
/usr/bin/oem-getlogs:    report[key + ".zip"] = cvalue
/usr/bin/oem-getlogs:    attach_pathglob_as_zip(report,
/usr/bin/oem-getlogs:    attach_pathglob_as_zip(report, ['/usr/share/alsa/ucm/*
/*',
/usr/bin/oem-getlogs:    attach_pathglob_as_zip(report, ['/sys/devices/*/*/drm/
card?/*/edid'],
/usr/bin/oem-getlogs:    attach_pathglob_as_zip(report,
/usr/bin/oem-getlogs:    attach_pathglob_as_zip(report, ["/var/log/*", "/var/lo
g/*/*"], "VAR_LOG")
/usr/bin/oem-getlogs:    attach_pathglob_as_zip(report, [
/usr/bin/oem-getlogs:    import gzip
```



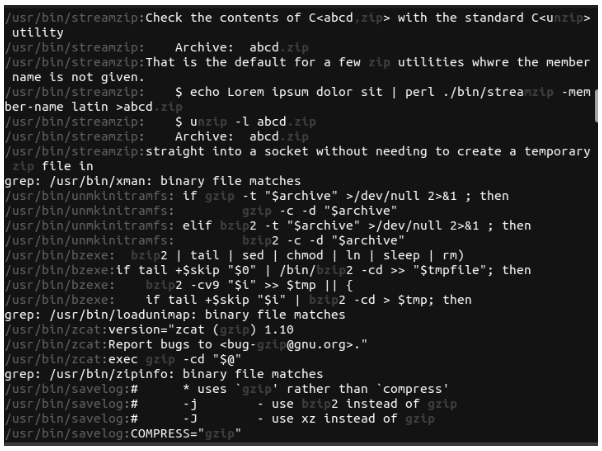
```
/usr/bin/oem-getlogs:    attach_pathglob_as_zip(report, [
/usr/bin/oem-getlogs:    import gzip
/usr/bin/oem-getlogs:    with gzip.open(filename, 'wb') as f:
grep: /usr/bin/mksquashfs: binary file matches
grep: /usr/bin/python3.10: binary file matches
grep: /usr/bin/rsync: binary file matches
grep: /usr/bin/bzip2: binary file matches
grep: /usr/bin/localedef: binary file matches
grep: /usr/bin/unzip: binary file matches
grep: /usr/bin/prezip-bin: binary file matches
/usr/bin/apport-cli:            if not hasattr(self.report[key], 'gzipvalue') a
nd \
/usr/bin/bzdiff:# Bzcmp/diff wrapped for bzip2,
/usr/bin/bzdiff:        bzip2 -cd "$FILE.bz2" | $comp $OPTIONS - "$FILE"
/usr/bin/bzdiff:                        bzip2 -cdfq "$2" > "$tmp"
/usr/bin/bzdiff:                        bzip2 -cdfq "$1" | $comp $OPTIONS - "$t
mp"
/usr/bin/bzdiff:                *)      bzip2 -cdfq "$1" | $comp $OPTIONS - "$2
"
/usr/bin/bzdiff:                        bzip2 -cdfq "$2" | $comp $OPTIONS "$1"
-
grep: /usr/bin/gnome-control-center: binary file matches
grep: /usr/bin/gnome-extensions: binary file matches
/usr/bin/uncompress:# Uncompress files.  This is the inverse of gzip.
/usr/bin/uncompress:version="gunzip (gzip) 1.10
/usr/bin/uncompress:Report bugs to <bug-gzip@gnu.org>."
/usr/bin/uncompress:exec gzip -d "$@"
grep: /usr/bin/file-roller: binary file matches
```



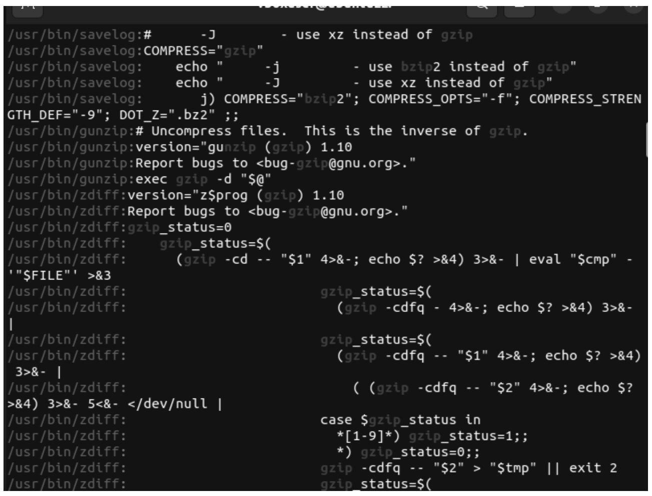
```
grep: /usr/bin/nautilus-sendto: binary file matches
grep: /usr/bin/apt-ftparchive: binary file matches
/usr/bin/zless:version="zless (gzip) 1.10
/usr/bin/zless:Report bugs to <bug-gzip@gnu.org>."
/usr/bin/zless:LESSOPEN="|$check_exit_status${use_input_pipe_on_stdin}gzip -cdf
q -- %s"
/usr/bin/streamzip:# Streaming zip
/usr/bin/streamzip:use IO::Compress::Zip qw(zip
/usr/bin/streamzip:my $zipfile = '-';
/usr/bin/streamzip:my $zip64 = 0 ;
/usr/bin/streamzip:GetOptions("zip64"          => \$zip64,
/usr/bin/streamzip:           "zipfile=s"      => \$zipfile,
/usr/bin/streamzip:zip '-' => $zipfile,
/usr/bin/streamzip:           Zip64  => $zip64,
/usr/bin/streamzip:    or die "Error creating zip file '$zipfile': $\n" ;
/usr/bin/streamzip:                  bzip2   => ZIP_CM_BZIP2,
/usr/bin/streamzip:  producer | streamzip [OPTIONS] | consumer
/usr/bin/streamzip:  producer | streamzip [OPTIONS] -zipfile output zip
/usr/bin/streamzip:  -zipfile=F      Write zip container to the filename 'F'
/usr/bin/streamzip:                  Outputs to stdout if zipfile not specified
.
/usr/bin/streamzip:  -zip64          Create a Zip64-compliant zip file [Default
: No]
/usr/bin/streamzip:  -stream         Force a streamed zip file when 'zipfile' o
ption is also enabled.
/usr/bin/streamzip:                  Only applies when 'zipfile' option is used
. [Default: No]
/usr/bin/streamzip:                    bzip2    Use Bzip2 compression
/usr/bin/streamzip:streamzip - create a zip file from stdin
```



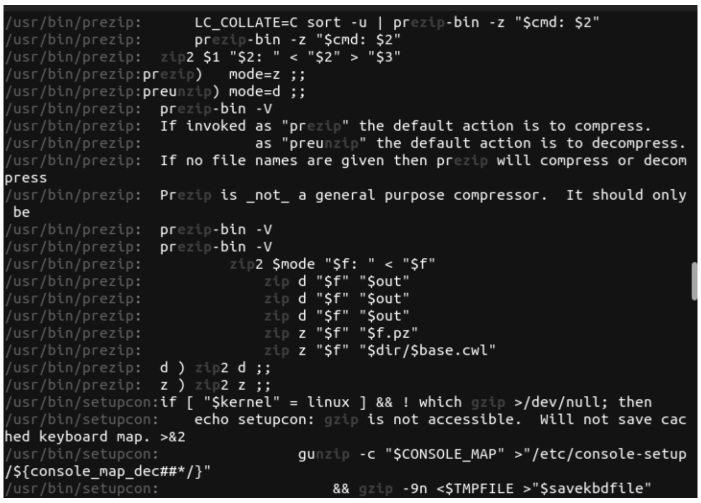
```
/usr/bin/streamzip:                    bzip2    Use Bzip2 compression
/usr/bin/streamzip:streamzip - create a zip file from stdin
/usr/bin/streamzip:    producer | streamzip [opts] | consumer
/usr/bin/streamzip:    producer | streamzip [opts] -zipfile=output.zip
/usr/bin/streamzip:This program will read data from C<stdin>, compress it into
a zip container
/usr/bin/streamzip:and, by default, write a I<streamed> zip file to C<stdout>.
No temporary
/usr/bin/streamzip:The zip container written to C<stdout> is, by necessity, wri
tten in
/usr/bin/streamzip:streamed zip file, but if interoperability is important, and
 your workflow
/usr/bin/streamzip:allows you to write the zip file directly to disk you can cr
eate a
/usr/bin/streamzip:non-streamed zip file using the C<zipfile> option.
/usr/bin/streamzip:=item -zip64
/usr/bin/streamzip:Create a Zip64-compliant zip container. Use this option if t
he input is
/usr/bin/streamzip:=item  -zipfile=F
/usr/bin/streamzip:Write zip container to the filename C<F>.
/usr/bin/streamzip:Use the C<Stream> option to force the creation of a streamed
 zip file.
/usr/bin/streamzip:This option is used to name the "file" in the zip container.
/usr/bin/streamzip:If the C<zipfile> option is specified, including this option
 will trigger
/usr/bin/streamzip:the creation of a streamed zip file.
/usr/bin/streamzip:    * bzip2    Use Bzip2 compression
/usr/bin/streamzip:Create a zip file bt reading daa from stdin
/usr/bin/streamzip:    $ echo Lorem ipsum dolor sit | perl ./bin/streamzip >abc
```



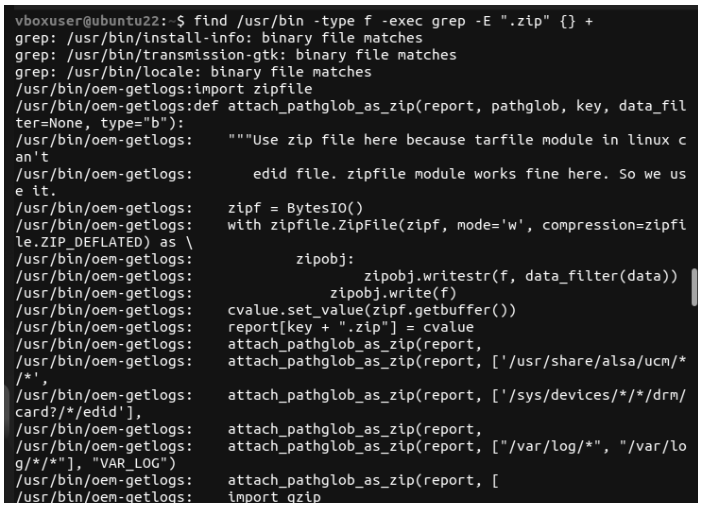
```
/usr/bin/streamzip:Check the contents of C<abcd,zip> with the standard C<unzip>
 utility
/usr/bin/streamzip:    Archive:  abcd.zip
/usr/bin/streamzip:That is the default for a few zip utilities whwre the member
 name is not given.
/usr/bin/streamzip:    $ echo Lorem ipsum dolor sit | perl ./bin/streamzip -mem
ber-name latin >abcd.zip
/usr/bin/streamzip:    $ unzip -l abcd.zip
/usr/bin/streamzip:    Archive:  abcd.zip
/usr/bin/streamzip:straight into a socket without needing to create a temporary
 zip file in
grep: /usr/bin/xman: binary file matches
/usr/bin/unmkinitramfs: if gzip -t "$archive" >/dev/null 2>&1 ; then
/usr/bin/unmkinitramfs:         gzip -c -d "$archive"
/usr/bin/unmkinitramfs: elif bzip2 -t "$archive" >/dev/null 2>&1 ; then
/usr/bin/unmkinitramfs:         bzip2 -c -d "$archive"
/usr/bin/bzexe:  bzip2 | tail | sed | chmod | ln | sleep | rm)
/usr/bin/bzexe:if tail +$skip "$0" | /bin/bzip2 -cd >> "$tmpfile"; then
/usr/bin/bzexe:    bzip2 -cv9 "$i" >> $tmp || {
/usr/bin/bzexe:    if tail +$skip "$i" | bzip2 -cd > $tmp; then
grep: /usr/bin/loadunimap: binary file matches
/usr/bin/zcat:version="zcat (gzip) 1.10
/usr/bin/zcat:Report bugs to <bug-gzip@gnu.org>."
/usr/bin/zcat:exec gzip -cd "$@"
grep: /usr/bin/zipinfo: binary file matches
/usr/bin/savelog:#      * uses `gzip' rather than `compress'
/usr/bin/savelog:#      -j         - use bzip2 instead of gzip
/usr/bin/savelog:#      -J         - use xz instead of gzip
/usr/bin/savelog:COMPRESS="gzip"
```



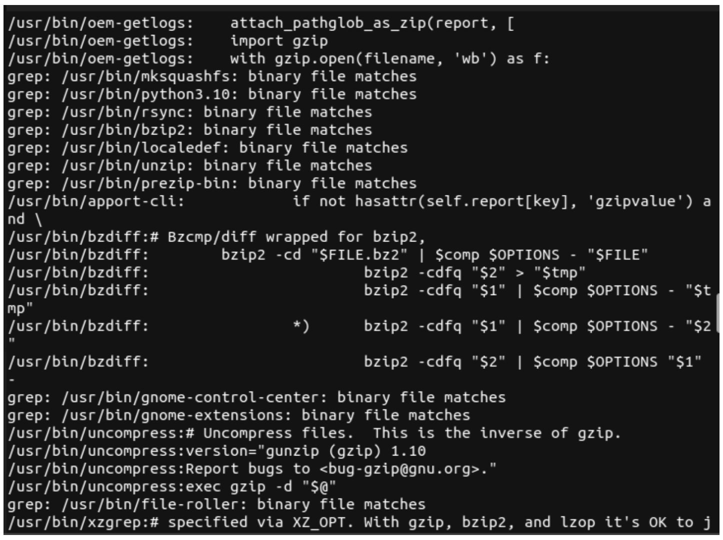
```
/usr/bin/savelog:#      -J         - use xz instead of gzip
/usr/bin/savelog:COMPRESS="gzip"
/usr/bin/savelog:    echo "     -j          - use bzip2 instead of gzip"
/usr/bin/savelog:    echo "     -J          - use xz instead of gzip"
/usr/bin/savelog:       j) COMPRESS="bzip2"; COMPRESS_OPTS="-f"; COMPRESS_STREN
GTH_DEF="-9"; DOT_Z=".bz2" ;;
/usr/bin/gunzip:# Uncompress files.  This is the inverse of gzip.
/usr/bin/gunzip:version="gunzip (gzip) 1.10
/usr/bin/gunzip:Report bugs to <bug-gzip@gnu.org>."
/usr/bin/gunzip:exec gzip -d "$@"
/usr/bin/zdiff:version="z$prog (gzip) 1.10
/usr/bin/zdiff:Report bugs to <bug-gzip@gnu.org>."
/usr/bin/zdiff:gzip_status=0
/usr/bin/zdiff:    gzip_status=$(
/usr/bin/zdiff:      (gzip -cd -- "$1" 4>&-; echo $? >&4) 3>&- | eval "$cmp" -
'"$FILE"' >&3
/usr/bin/zdiff:                         gzip_status=$(
/usr/bin/zdiff:                           (gzip -cdfq - 4>&-; echo $? >&4) 3>&-
|
/usr/bin/zdiff:                         gzip_status=$(
/usr/bin/zdiff:                           (gzip -cdfq -- "$1" 4>&-; echo $? >&4)
 3>&- |
/usr/bin/zdiff:                             ( (gzip -cdfq -- "$2" 4>&-; echo $?
>&4) 3>&- 5<&- </dev/null |
/usr/bin/zdiff:                         case $gzip_status in
/usr/bin/zdiff:                           *[1-9]*) gzip_status=1;;
/usr/bin/zdiff:                           *) gzip_status=0;;
/usr/bin/zdiff:                         gzip -cdfq -- "$2" > "$tmp" || exit 2
/usr/bin/zdiff:                         gzip_status=$(
```



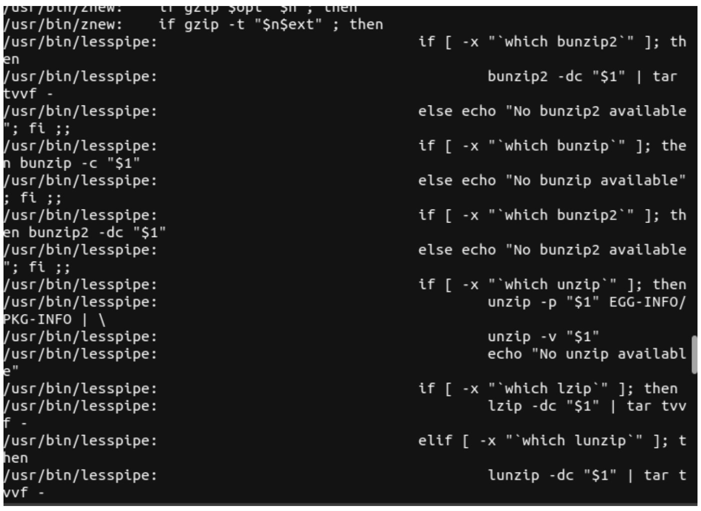
```
/usr/bin/prezip:      LC_COLLATE=C sort -u | prezip-bin -z "$cmd: $2"
/usr/bin/prezip:      prezip-bin -z "$cmd: $2"
/usr/bin/prezip:  zip2 $1 "$2: " < "$2" > "$3"
/usr/bin/prezip:prezip)   mode=z ;;
/usr/bin/prezip:preunzip) mode=d ;;
/usr/bin/prezip:  prezip-bin -V
/usr/bin/prezip:  If invoked as "prezip" the default action is to compress.
/usr/bin/prezip:             as "preunzip" the default action is to decompress.
/usr/bin/prezip:  If no file names are given then prezip will compress or decom
press
/usr/bin/prezip:  Prezip is _not_ a general purpose compressor.  It should only
 be
/usr/bin/prezip:  prezip-bin -V
/usr/bin/prezip:  prezip-bin -V
/usr/bin/prezip:          zip2 $mode "$f: " < "$f"
/usr/bin/prezip:              zip d "$f" "$out"
/usr/bin/prezip:              zip d "$f" "$out"
/usr/bin/prezip:              zip d "$f" "$out"
/usr/bin/prezip:              zip z "$f" "$f.pz"
/usr/bin/prezip:              zip z "$f" "$dir/$base.cwl"
/usr/bin/prezip:  d ) zip2 d ;;
/usr/bin/prezip:  z ) zip2 z ;;
/usr/bin/setupcon:if [ "$kernel" = linux ] && ! which gzip >/dev/null; then
/usr/bin/setupcon:    echo setupcon: gzip is not accessible.  Will not save cac
hed keyboard map. >&2
/usr/bin/setupcon:                 gunzip -c "$CONSOLE_MAP" >"/etc/console-setup
/${console_map_dec##*/}"
/usr/bin/setupcon:                     && gzip -9n <$TMPFILE >"$savekbdfile"
```



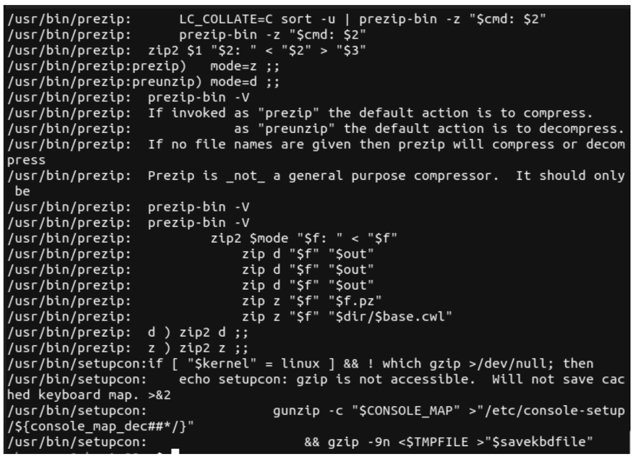
```
vboxuser@ubuntu22:~$ find /usr/bin -type f -exec grep -E ".zip" {} +
grep: /usr/bin/install-info: binary file matches
grep: /usr/bin/transmission-gtk: binary file matches
grep: /usr/bin/locale: binary file matches
/usr/bin/oem-getlogs:import zipfile
/usr/bin/oem-getlogs:def attach_pathglob_as_zip(report, pathglob, key, data_fil
ter=None, type="b"):
/usr/bin/oem-getlogs:    """Use zip file here because tarfile module in linux c
an't
/usr/bin/oem-getlogs:       edid file. zipfile module works fine here. So we us
e it.
/usr/bin/oem-getlogs:    zipf = BytesIO()
/usr/bin/oem-getlogs:    with zipfile.ZipFile(zipf, mode='w', compression=zipfi
le.ZIP_DEFLATED) as \
/usr/bin/oem-getlogs:            zipobj:
/usr/bin/oem-getlogs:                    zipobj.writestr(f, data_filter(data))
/usr/bin/oem-getlogs:                zipobj.write(f)
/usr/bin/oem-getlogs:    cvalue.set_value(zipf.getbuffer())
/usr/bin/oem-getlogs:    report[key + ".zip"] = cvalue
/usr/bin/oem-getlogs:    attach_pathglob_as_zip(report,
/usr/bin/oem-getlogs:    attach_pathglob_as_zip(report, ['/usr/share/alsa/ucm/*
/*',
/usr/bin/oem-getlogs:    attach_pathglob_as_zip(report, ['/sys/devices/*/*/drm/
card?/*/edid'],
/usr/bin/oem-getlogs:    attach_pathglob_as_zip(report,
/usr/bin/oem-getlogs:    attach_pathglob_as_zip(report, ["/var/log/*", "/var/lo
g/*/*"], "VAR_LOG")
/usr/bin/oem-getlogs:    attach_pathglob_as_zip(report, [
/usr/bin/oem-getlogs:    import gzip
```



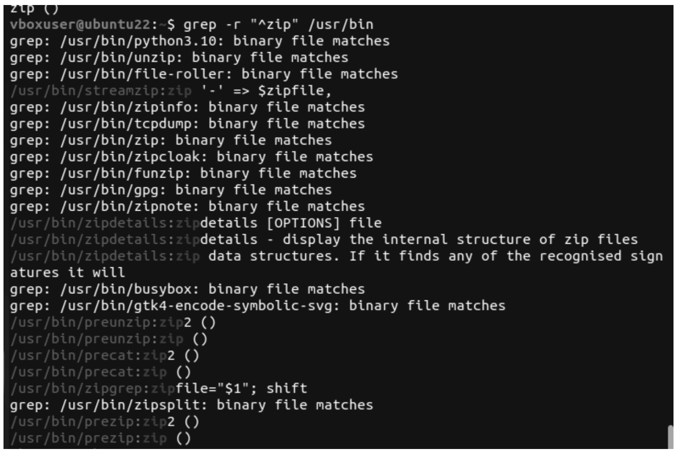
```
/usr/bin/oem-getlogs:    attach_pathglob_as_zip(report, [
/usr/bin/oem-getlogs:    import gzip
/usr/bin/oem-getlogs:    with gzip.open(filename, 'wb') as f:
grep: /usr/bin/mksquashfs: binary file matches
grep: /usr/bin/python3.10: binary file matches
grep: /usr/bin/rsync: binary file matches
grep: /usr/bin/bzip2: binary file matches
grep: /usr/bin/localedef: binary file matches
grep: /usr/bin/unzip: binary file matches
grep: /usr/bin/prezip-bin: binary file matches
/usr/bin/apport-cli:            if not hasattr(self.report[key], 'gzipvalue') a
nd \
/usr/bin/bzdiff:# Bzcmp/diff wrapped for bzip2,
/usr/bin/bzdiff:        bzip2 -cd "$FILE.bz2" | $comp $OPTIONS - "$FILE"
/usr/bin/bzdiff:                        bzip2 -cdfq "$2" > "$tmp"
/usr/bin/bzdiff:                        bzip2 -cdfq "$1" | $comp $OPTIONS - "$t
mp"
/usr/bin/bzdiff:                *)      bzip2 -cdfq "$1" | $comp $OPTIONS - "$2
"
/usr/bin/bzdiff:                        bzip2 -cdfq "$2" | $comp $OPTIONS "$1"
-
grep: /usr/bin/gnome-control-center: binary file matches
grep: /usr/bin/gnome-extensions: binary file matches
/usr/bin/uncompress:# Uncompress files.  This is the inverse of gzip.
/usr/bin/uncompress:version="gunzip (gzip) 1.10
/usr/bin/uncompress:Report bugs to <bug-gzip@gnu.org>."
/usr/bin/uncompress:exec gzip -d "$@"
grep: /usr/bin/file-roller: binary file matches
/usr/bin/xzgrep:# specified via XZ_OPT. With gzip, bzip2, and lzop it's OK to j
```



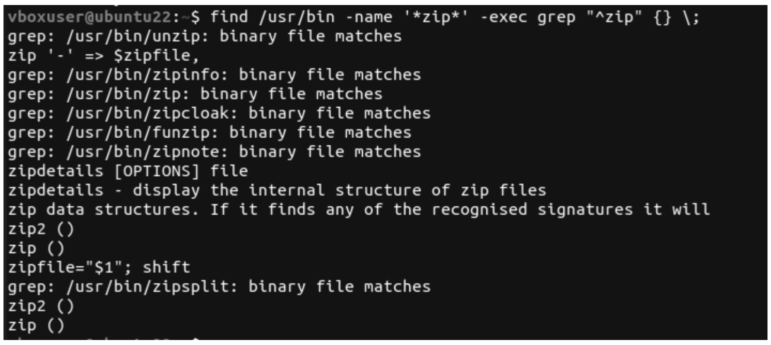
```
/usr/bin/znew:    if gzip $opt "$n" ; then
/usr/bin/znew:    if gzip -t "$n$ext" ; then
/usr/bin/lesspipe:                              if [ -x "`which bunzip2`" ]; th
en
/usr/bin/lesspipe:                                      bunzip2 -dc "$1" | tar
tvvf -
/usr/bin/lesspipe:                              else echo "No bunzip2 available
"; fi ;;
/usr/bin/lesspipe:                              if [ -x "`which bunzip`" ]; the
n bunzip -c "$1"
/usr/bin/lesspipe:                              else echo "No bunzip available"
; fi ;;
/usr/bin/lesspipe:                              if [ -x "`which bunzip2`" ]; th
en bunzip2 -dc "$1"
/usr/bin/lesspipe:                              else echo "No bunzip2 available
"; fi ;;
/usr/bin/lesspipe:                              if [ -x "`which unzip`" ]; then
/usr/bin/lesspipe:                                      unzip -p "$1" EGG-INFO/
PKG-INFO | \
/usr/bin/lesspipe:                                      unzip -v "$1"
/usr/bin/lesspipe:                                      echo "No unzip availabl
e"
/usr/bin/lesspipe:                              if [ -x "`which lzip`" ]; then
/usr/bin/lesspipe:                                      lzip -dc "$1" | tar tvv
f -
/usr/bin/lesspipe:                              elif [ -x "`which lunzip`" ]; t
hen
/usr/bin/lesspipe:                                      lunzip -dc "$1" | tar t
vvf -
```



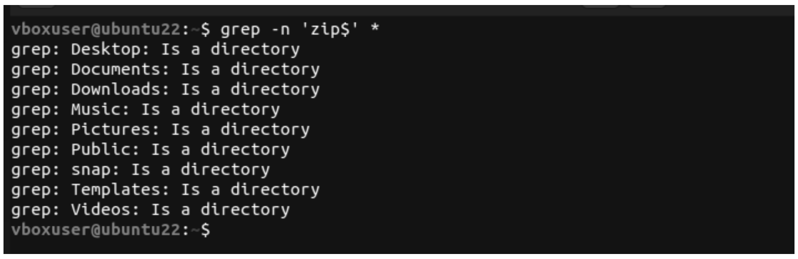
```
/usr/bin/prezip:      LC_COLLATE=C sort -u | prezip-bin -z "$cmd: $2"
/usr/bin/prezip:      prezip-bin -z "$cmd: $2"
/usr/bin/prezip:  zip2 $1 "$2: " < "$2" > "$3"
/usr/bin/prezip:prezip)   mode=z ;;
/usr/bin/prezip:preunzip) mode=d ;;
/usr/bin/prezip:  prezip-bin -V
/usr/bin/prezip:  If invoked as "prezip" the default action is to compress.
/usr/bin/prezip:             as "preunzip" the default action is to decompress.
/usr/bin/prezip:  If no file names are given then prezip will compress or decom
press
/usr/bin/prezip:  Prezip is _not_ a general purpose compressor.  It should only
 be
/usr/bin/prezip:  prezip-bin -V
/usr/bin/prezip:  prezip-bin -V
/usr/bin/prezip:          zip2 $mode "$f: " < "$f"
/usr/bin/prezip:              zip d "$f" "$out"
/usr/bin/prezip:              zip d "$f" "$out"
/usr/bin/prezip:              zip d "$f" "$out"
/usr/bin/prezip:              zip z "$f" "$f.pz"
/usr/bin/prezip:              zip z "$f" "$dir/$base.cwl"
/usr/bin/prezip:  d ) zip2 d ;;
/usr/bin/prezip:  z ) zip2 z ;;
/usr/bin/setupcon:if [ "$kernel" = linux ] && ! which gzip >/dev/null; then
/usr/bin/setupcon:    echo setupcon: gzip is not accessible.  Will not save cac
hed keyboard map. >&2
/usr/bin/setupcon:                gunzip -c "$CONSOLE_MAP" >"/etc/console-setup
/${console_map_dec##*/}"
/usr/bin/setupcon:                    && gzip -9n <$TMPFILE >"$savekbdfile"
```



```
zip ()
vboxuser@ubuntu22:~$ grep -r "^zip" /usr/bin
grep: /usr/bin/python3.10: binary file matches
grep: /usr/bin/unzip: binary file matches
grep: /usr/bin/file-roller: binary file matches
/usr/bin/streamzip:zip '-' => $zipfile,
grep: /usr/bin/zipinfo: binary file matches
grep: /usr/bin/tcpdump: binary file matches
grep: /usr/bin/zip: binary file matches
grep: /usr/bin/zipcloak: binary file matches
grep: /usr/bin/funzip: binary file matches
grep: /usr/bin/gpg: binary file matches
grep: /usr/bin/zipnote: binary file matches
/usr/bin/zipdetails:zipdetails [OPTIONS] file
/usr/bin/zipdetails:zipdetails - display the internal structure of zip files
/usr/bin/zipdetails:zip data structures. If it finds any of the recognised sign
atures it will
grep: /usr/bin/busybox: binary file matches
grep: /usr/bin/gtk4-encode-symbolic-svg: binary file matches
/usr/bin/preunzip:zip2 ()
/usr/bin/preunzip:zip ()
/usr/bin/precat:zip2 ()
/usr/bin/precat:zip ()
/usr/bin/zipgrep:zipfile="$1"; shift
grep: /usr/bin/zipsplit: binary file matches
/usr/bin/prezip:zip2 ()
/usr/bin/prezip:zip ()
```



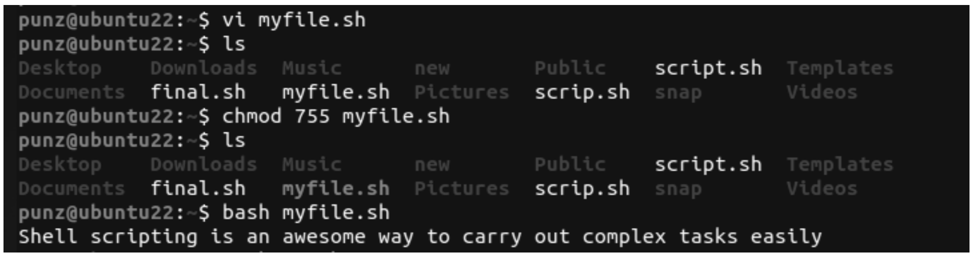
```
vboxuser@ubuntu22:~$ find /usr/bin -name '*zip*' -exec grep "^zip" {} \;
grep: /usr/bin/unzip: binary file matches
zip '-' => $zipfile,
grep: /usr/bin/zipinfo: binary file matches
grep: /usr/bin/zip: binary file matches
grep: /usr/bin/zipcloak: binary file matches
grep: /usr/bin/funzip: binary file matches
grep: /usr/bin/zipnote: binary file matches
zipdetails [OPTIONS] file
zipdetails - display the internal structure of zip files
zip data structures. If it finds any of the recognised signatures it will
zip2 ()
zip ()
zipfile="$1"; shift
grep: /usr/bin/zipsplit: binary file matches
zip2 ()
zip ()
```



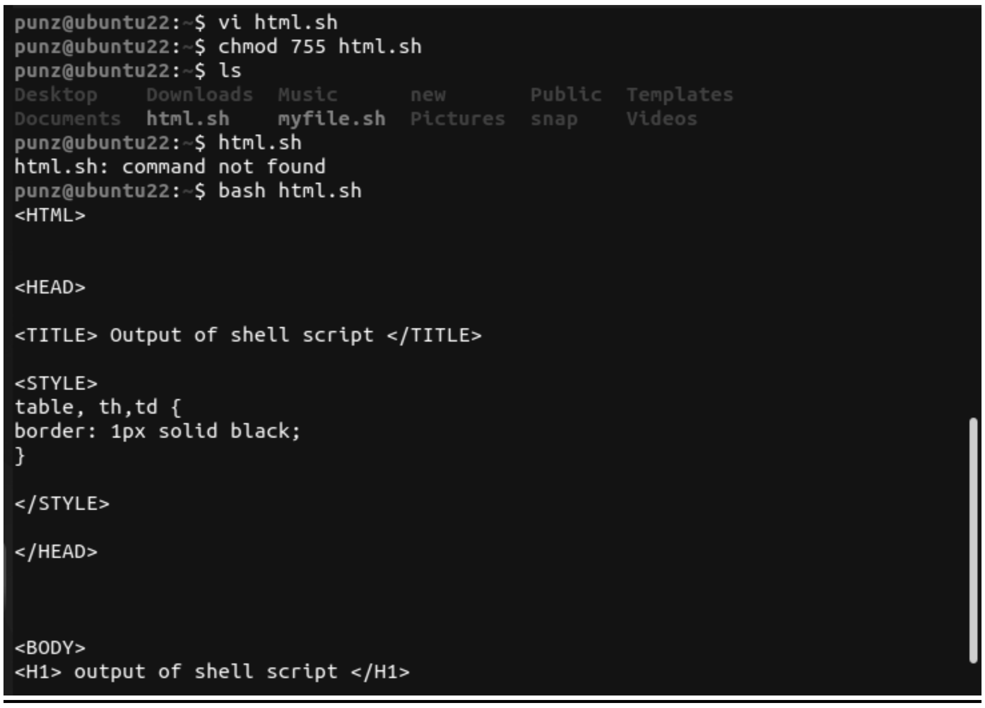
```
vboxuser@ubuntu22:~$ grep -n 'zip$' *
grep: Desktop: Is a directory
grep: Documents: Is a directory
grep: Downloads: Is a directory
grep: Music: Is a directory
grep: Pictures: Is a directory
grep: Public: Is a directory
grep: snap: Is a directory
grep: Templates: Is a directory
grep: Videos: Is a directory
vboxuser@ubuntu22:~$
```

E21CSEU0760

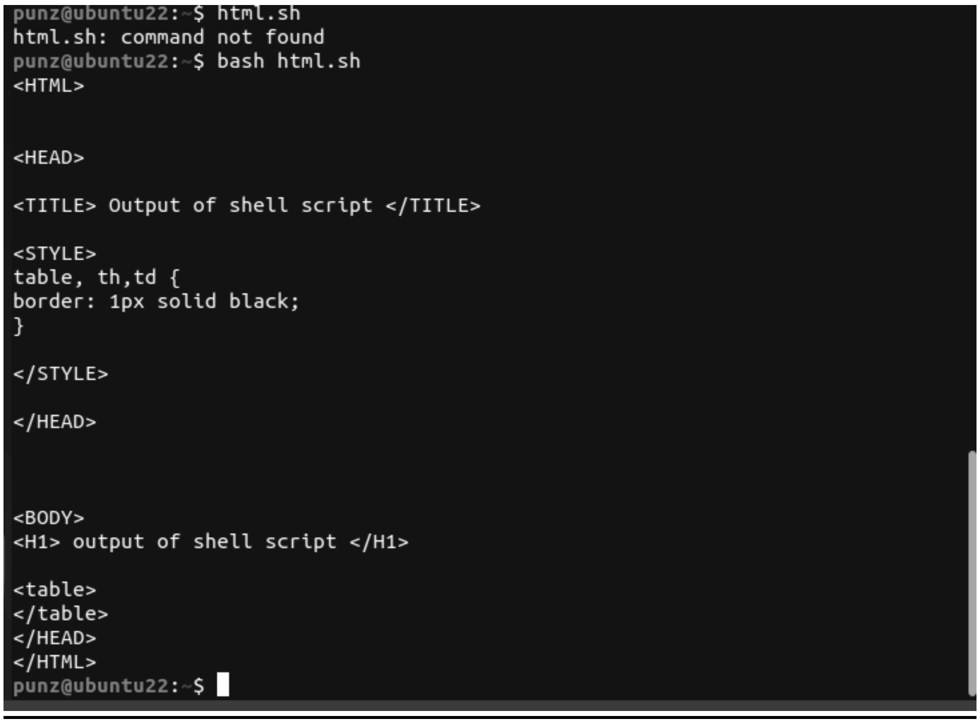
LAB 3



```
punz@ubuntu22:~$ vi myfile.sh
punz@ubuntu22:~$ ls
Desktop    Downloads  Music     new       Public    script.sh  Templates
Documents  final.sh   myfile.sh Pictures  scrip.sh  snap       Videos
punz@ubuntu22:~$ chmod 755 myfile.sh
punz@ubuntu22:~$ ls
Desktop    Downloads  Music     new       Public    script.sh  Templates
Documents  final.sh   myfile.sh Pictures  scrip.sh  snap       Videos
punz@ubuntu22:~$ bash myfile.sh
Shell scripting is an awesome way to carry out complex tasks easily
```



```
punz@ubuntu22:~$ vi html.sh
punz@ubuntu22:~$ chmod 755 html.sh
punz@ubuntu22:~$ ls
Desktop    Downloads  Music      new       Public  Templates
Documents  html.sh    myfile.sh  Pictures  snap    Videos
punz@ubuntu22:~$ html.sh
html.sh: command not found
punz@ubuntu22:~$ bash html.sh
<HTML>


<HEAD>

<TITLE> Output of shell script </TITLE>

<STYLE>
table, th,td {
border: 1px solid black;
}

</STYLE>

</HEAD>



<BODY>
<H1> output of shell script </H1>
```

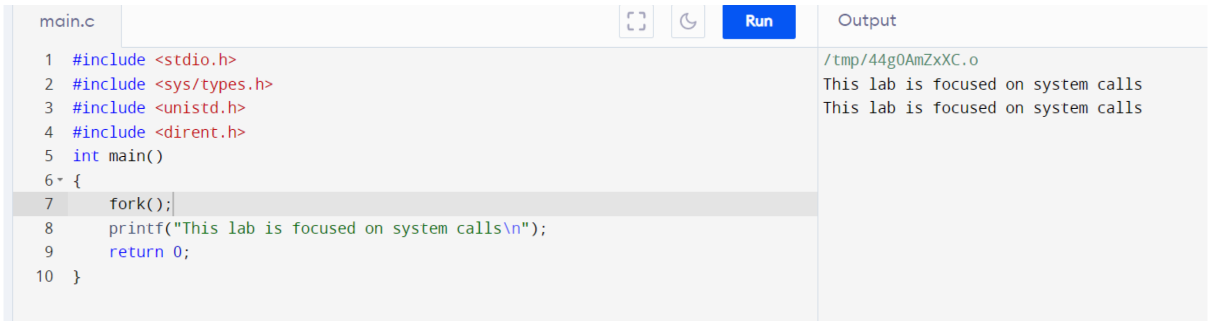


```
punz@ubuntu22:~$ html.sh
html.sh: command not found
punz@ubuntu22:~$ bash html.sh
<HTML>


<HEAD>

<TITLE> Output of shell script </TITLE>

<STYLE>
table, th,td {
border: 1px solid black;
}

</STYLE>

</HEAD>



<BODY>
<H1> output of shell script </H1>

<table>
</table>
</HEAD>
</HTML>
punz@ubuntu22:~$
```

LAB 4

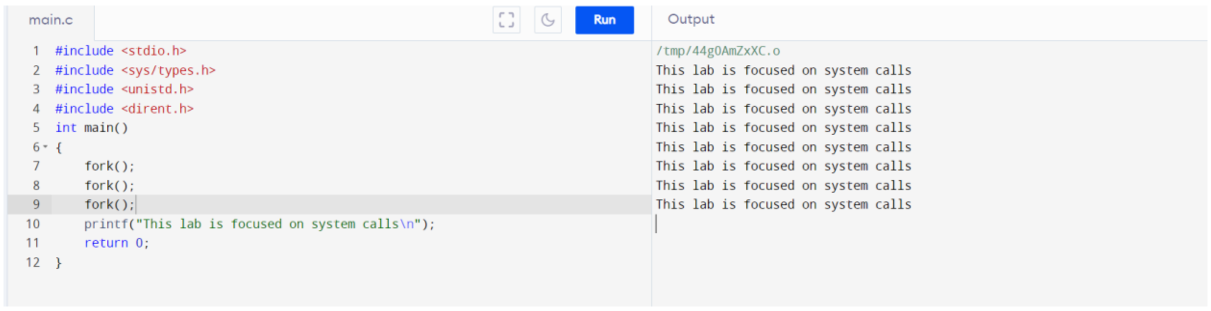
Soumya Dubey

E21CSEU0760



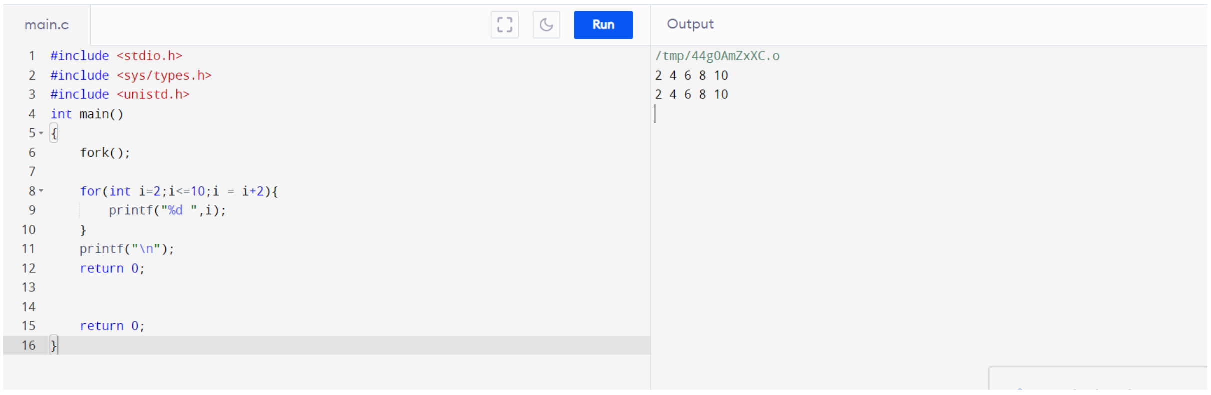
```c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <dirent.h>
int main()
{
    fork();
    printf("This lab is focused on system calls\n");
    return 0;
}
```

```
/tmp/44g0AmZxXC.o
This lab is focused on system calls
This lab is focused on system calls
```



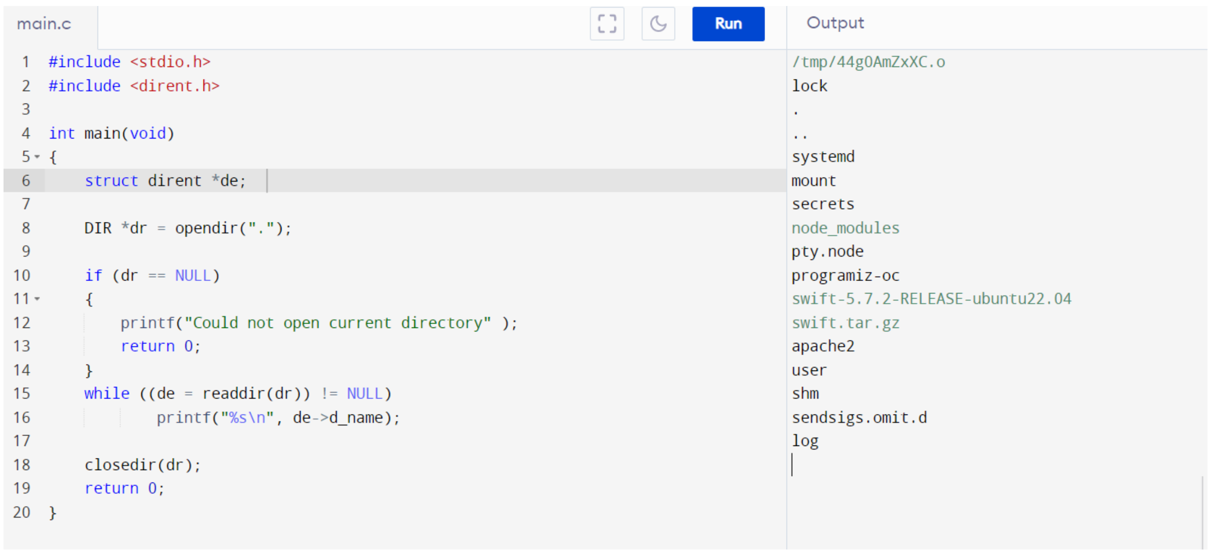
```c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <dirent.h>
int main()
{
    fork();
    fork();
    fork();
    printf("This lab is focused on system calls\n");
    return 0;
}
```

```
/tmp/44g0AmZxXC.o
This lab is focused on system calls
This lab is focused on system calls
This lab is focused on system calls
This lab is focused on system calls
This lab is focused on system calls
This lab is focused on system calls
This lab is focused on system calls
This lab is focused on system calls
```



```c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main()
{
    fork();

    for(int i=2;i<=10;i = i+2){
        printf("%d ",i);
    }
    printf("\n");
    return 0;


    return 0;
}
```

```
/tmp/44g0AmZxXC.o
2 4 6 8 10
2 4 6 8 10
```



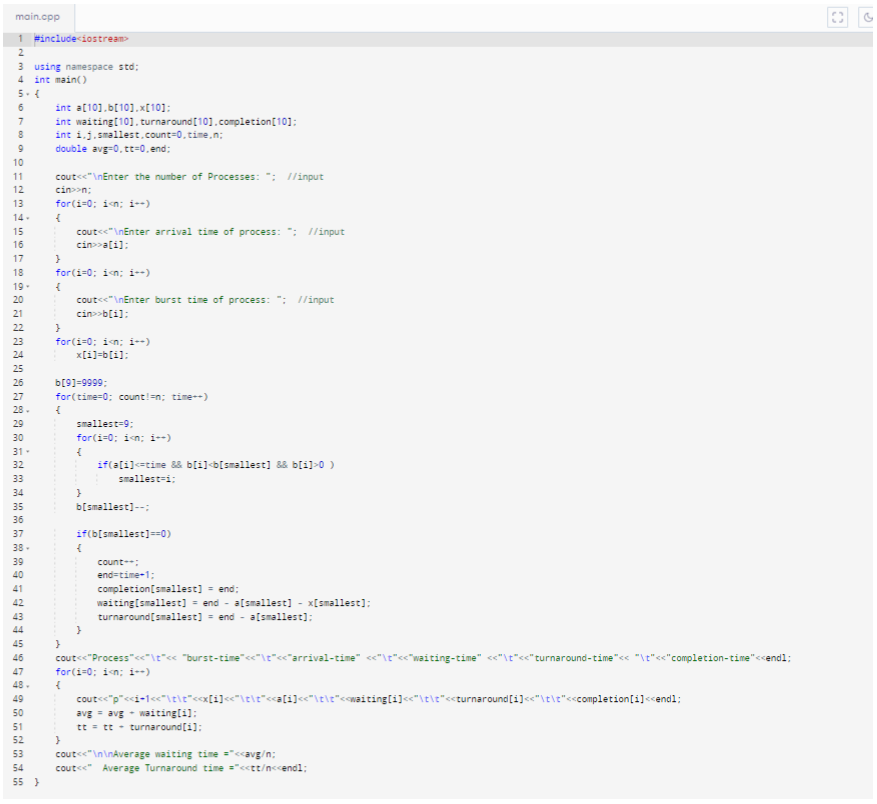
```c
#include <stdio.h>
#include <dirent.h>

int main(void)
{
    struct dirent *de;

    DIR *dr = opendir(".");

    if (dr == NULL)
    {
        printf("Could not open current directory" );
        return 0;
    }
    while ((de = readdir(dr)) != NULL)
            printf("%s\n", de->d_name);

    closedir(dr);
    return 0;
}
```

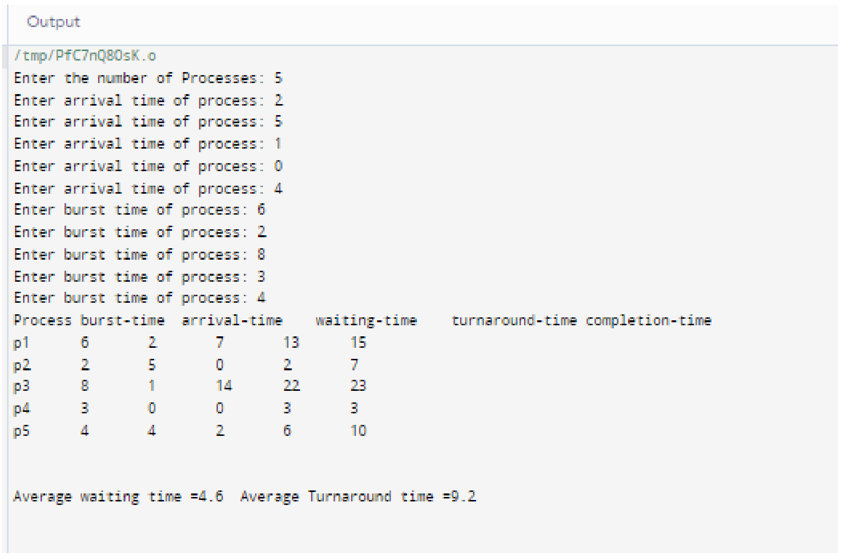
```
/tmp/44g0AmZxXC.o
lock
.
..
systemd
mount
secrets
node_modules
pty.node
programiz-oc
swift-5.7.2-RELEASE-ubuntu22.04
swift.tar.gz
apache2
user
shm
sendsigs.omit.d
log
```

Lab 06

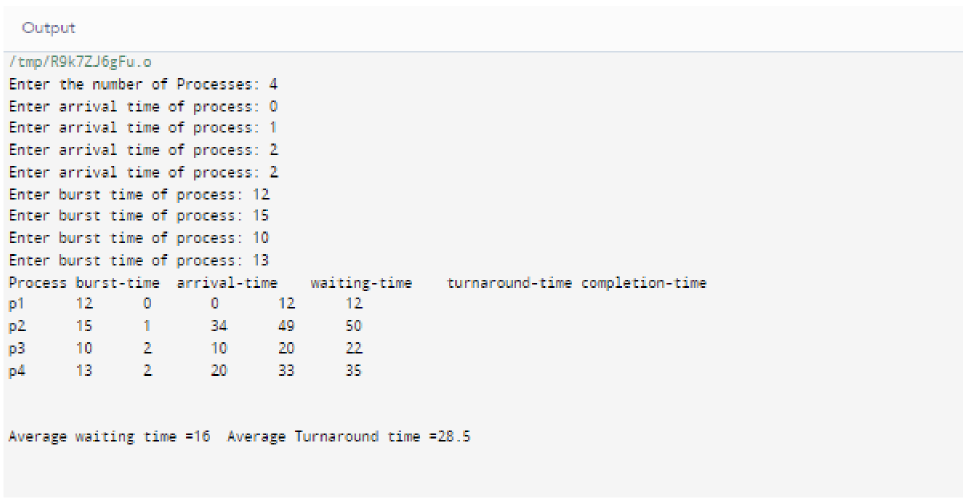
Soumya Dubey

E21CSEU0760


```
main.cpp
#include<iostream>

using namespace std;
int main()
{
    int a[10],b[10],x[10];
    int waiting[10],turnaround[10],completion[10];
    int i,j,smallest,count=0,time,n;
    double avg=0,tt=0,end;

    cout<<"\nEnter the number of Processes: ";  //input
    cin>>n;
    for(i=0; i<n; i++)
    {
        cout<<"\nEnter arrival time of process: ";  //input
        cin>>a[i];
    }
    for(i=0; i<n; i++)
    {
        cout<<"\nEnter burst time of process: ";  //input
        cin>>b[i];
    }
    for(i=0; i<n; i++)
        x[i]=b[i];

    b[9]=9999;
    for(time=0; count!=n; time++)
    {
        smallest=9;
        for(i=0; i<n; i++)
        {
            if(a[i]<=time && b[i]<b[smallest] && b[i]>0 )
                smallest=i;
        }
        b[smallest]--;

        if(b[smallest]==0)
        {
            count++;
            end=time+1;
            completion[smallest] = end;
            waiting[smallest] = end - a[smallest] - x[smallest];
            turnaround[smallest] = end - a[smallest];
        }
    }
    cout<<"Process"<<"\t"<< "burst-time"<<"\t"<<"arrival-time" <<"\t"<<"waiting-time" <<"\t"<<"turnaround-time"<< "\t"<<"completion-time"<<endl;
    for(i=0; i<n; i++)
    {
        cout<<"p"<<i+1<<"\t\t"<<x[i]<<"\t\t"<<a[i]<<"\t\t"<<waiting[i]<<"\t\t"<<turnaround[i]<<"\t\t"<<completion[i]<<endl;
        avg = avg + waiting[i];
        tt = tt + turnaround[i];
    }
    cout<<"\n\nAverage waiting time ="<<avg/n;
    cout<<"  Average Turnaround time ="<<tt/n<<endl;
}
```

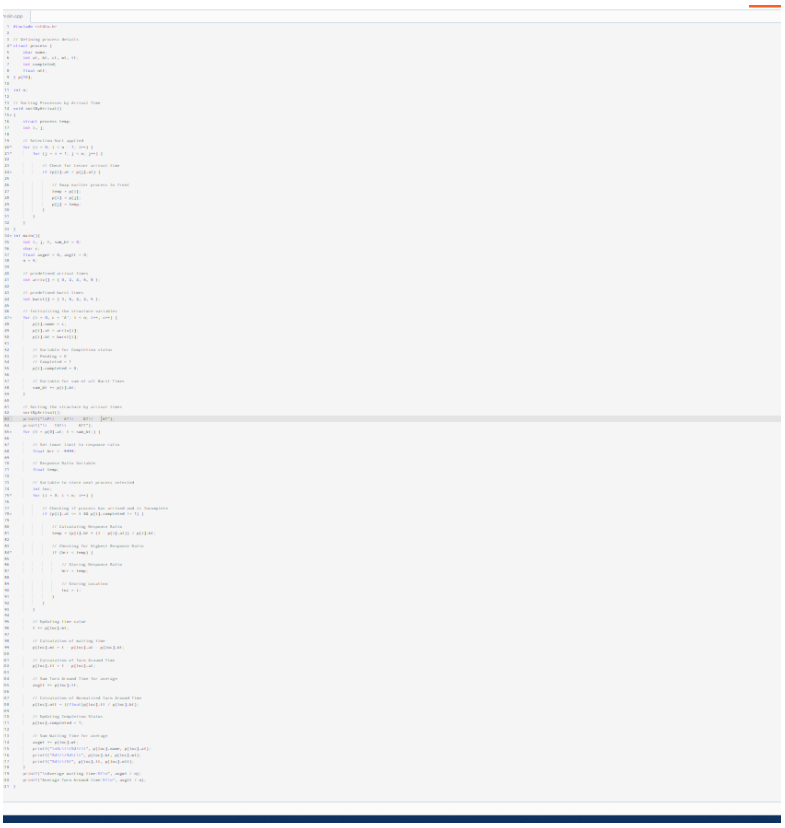

Output

```
/tmp/PfC7nQ80sK.o
Enter the number of Processes: 5
Enter arrival time of process: 2
Enter arrival time of process: 5
Enter arrival time of process: 1
Enter arrival time of process: 0
Enter arrival time of process: 4
Enter burst time of process: 6
Enter burst time of process: 2
Enter burst time of process: 8
Enter burst time of process: 3
Enter burst time of process: 4
Process burst-time  arrival-time   waiting-time    turnaround-time completion-time
p1      6       2       7       13      15
p2      2       5       0       2       7
p3      8       1       14      22      23
p4      3       0       0       3       3
p5      4       4       2       6       10


Average waiting time =4.6  Average Turnaround time =9.2
```

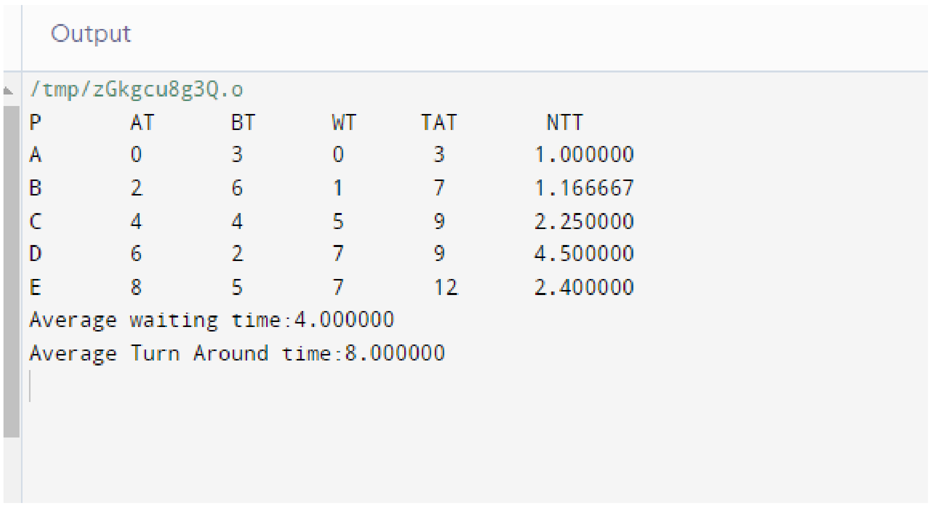


```
/tmp/R9k7ZJ6gFu.o
Enter the number of Processes: 4
Enter arrival time of process: 0
Enter arrival time of process: 1
Enter arrival time of process: 2
Enter arrival time of process: 2
Enter burst time of process: 12
Enter burst time of process: 15
Enter burst time of process: 10
Enter burst time of process: 13
Process burst-time  arrival-time    waiting-time    turnaround-time completion-time
p1      12      0       0       12      12
p2      15      1       34      49      50
p3      10      2       10      20      22
p4      13      2       20      33      35


Average waiting time =16  Average Turnaround time =28.5
```

TASK 2



```
#include <stdio.h>

// Defining process details
struct process {
    char name;
    int at, bt, ct, wt, tt;
    int completed;
    float ntt;
} p[10];

int n;

// Sorting Processes by Arrival Time
void sortByArrival()
{
    struct process temp;
    int i, j;

    // Selection Sort applied
    for (i = 0; i < n - 1; i++) {
        for (j = i + 1; j < n; j++) {

            // Check for lesser arrival time
            if (p[i].at > p[j].at) {

                // Swap earlier process to front
                temp = p[i];
                p[i] = p[j];
                p[j] = temp;
            }
        }
    }
}
int main(){
    int i, j, t, sum_bt = 0;
    char c;
    float avgwt = 0, avgtt = 0;
    n = 5;

    // predefined arrival times
    int arriv[] = { 0, 2, 4, 6, 8 };

    // predefined burst times
    int burst[] = { 3, 6, 4, 2, 5 };

    // Initializing the structure variables
    for (i = 0, c = 'A'; i < n; i++, c++) {
        p[i].name = c;
        p[i].at = arriv[i];
        p[i].bt = burst[i];

        // Variable for Completion status
        // Pending = 0
        // Completed = 1
        p[i].completed = 0;

        // Variable for sum of all Burst Times
        sum_bt += p[i].bt;
    }

    // Sorting the structure by arrival times
    sortByArrival();
    printf("\nPN\t    AT\t    BT\t    WT");
    printf("\t   TAT\t     NTT");
    for (t = p[0].at; t < sum_bt;) {

        // Set lower limit to response ratio
        float hrr = -9999;

        // Response Ratio Variable
        float temp;

        // Variable to store next process selected
        int loc;
        for (i = 0; i < n; i++) {

            // Checking if process has arrived and is Incomplete
            if (p[i].at <= t && p[i].completed != 1) {

                // Calculating Response Ratio
                temp = (p[i].bt + (t - p[i].at)) / p[i].bt;

                // Checking for Highest Response Ratio
                if (hrr < temp) {

                    // Storing Response Ratio
                    hrr = temp;

                    // Storing Location
                    loc = i;
                }
            }
        }

        // Updating time value
        t += p[loc].bt;

        // Calculation of waiting time
        p[loc].wt = t - p[loc].at - p[loc].bt;

        // Calculation of Turn Around Time
        p[loc].tt = t - p[loc].at;

        // Sum Turn Around Time for average
        avgtt += p[loc].tt;

        // Calculation of Normalized Turn Around Time
        p[loc].ntt = ((float)p[loc].tt / p[loc].bt);

        // Updating Completion Status
        p[loc].completed = 1;

        // Sum Waiting Time for average
        avgwt += p[loc].wt;
        printf("\n%c\t\t%d\t\t", p[loc].name, p[loc].at);
        printf("%d\t\t%d\t\t", p[loc].bt, p[loc].wt);
        printf("%d\t\t%f", p[loc].tt, p[loc].ntt);
    }
    printf("\nAverage waiting time:%f\n", avgwt / n);
    printf("Average Turn Around time:%f\n", avgtt / n);
}
```

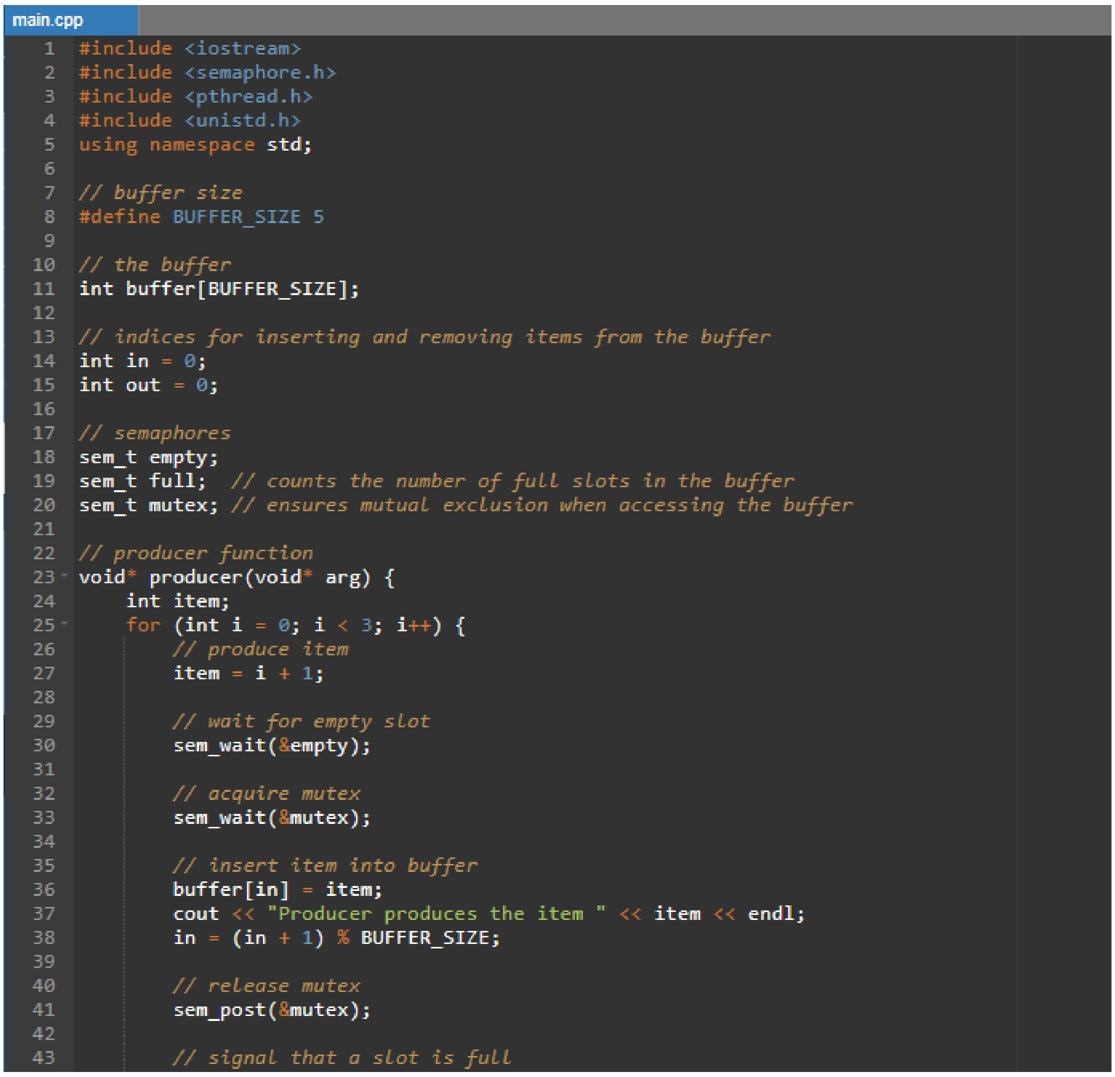

Output

```
/tmp/zGkgcu8g3Q.o
P        AT       BT       WT      TAT        NTT
A        0        3        0        3        1.000000
B        2        6        1        7        1.166667
C        4        4        5        9        2.250000
D        6        2        7        9        4.500000
E        8        5        7        12       2.400000
Average waiting time:4.000000
Average Turn Around time:8.000000
```

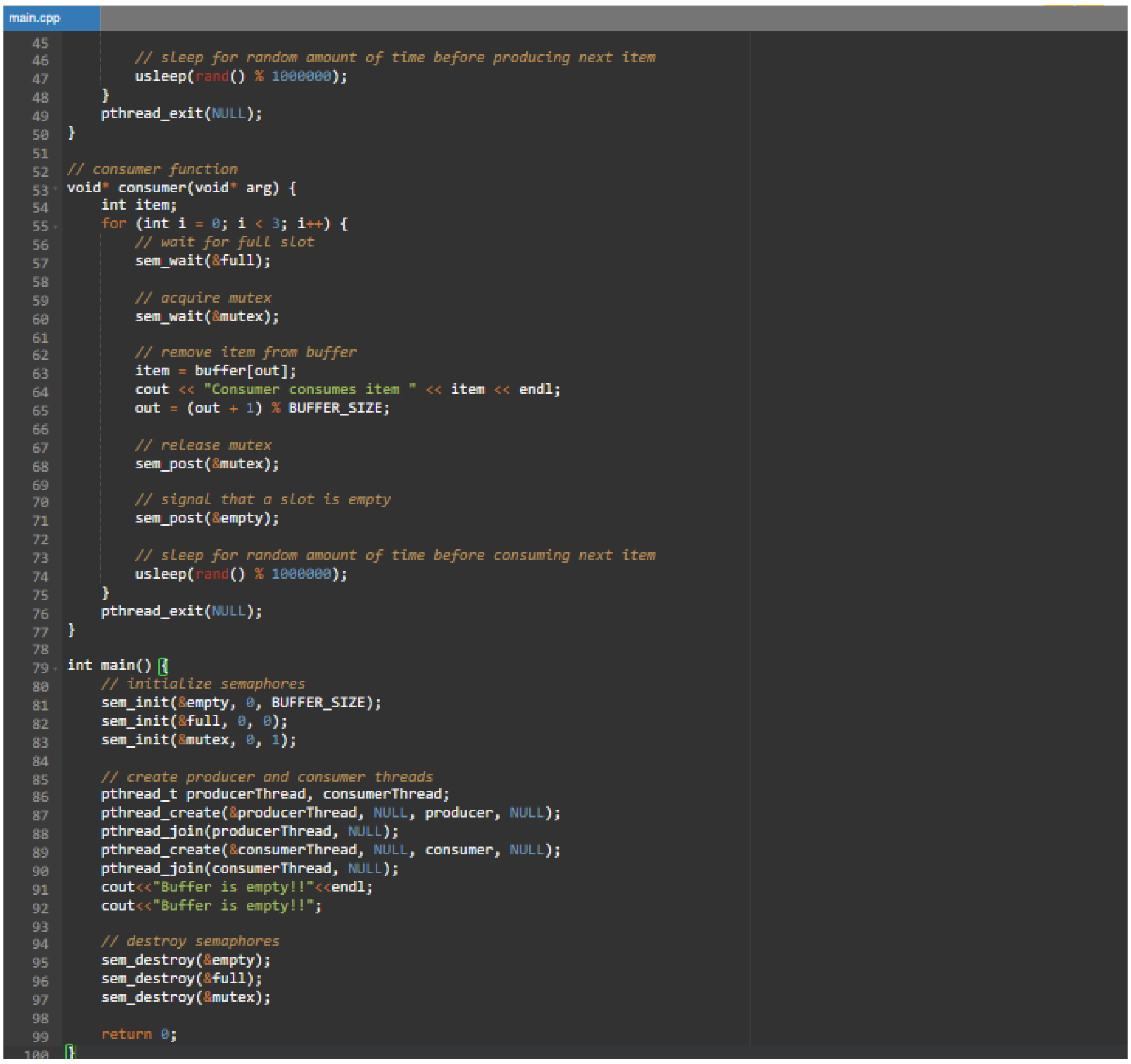
LAB 07

Soumya Dubey

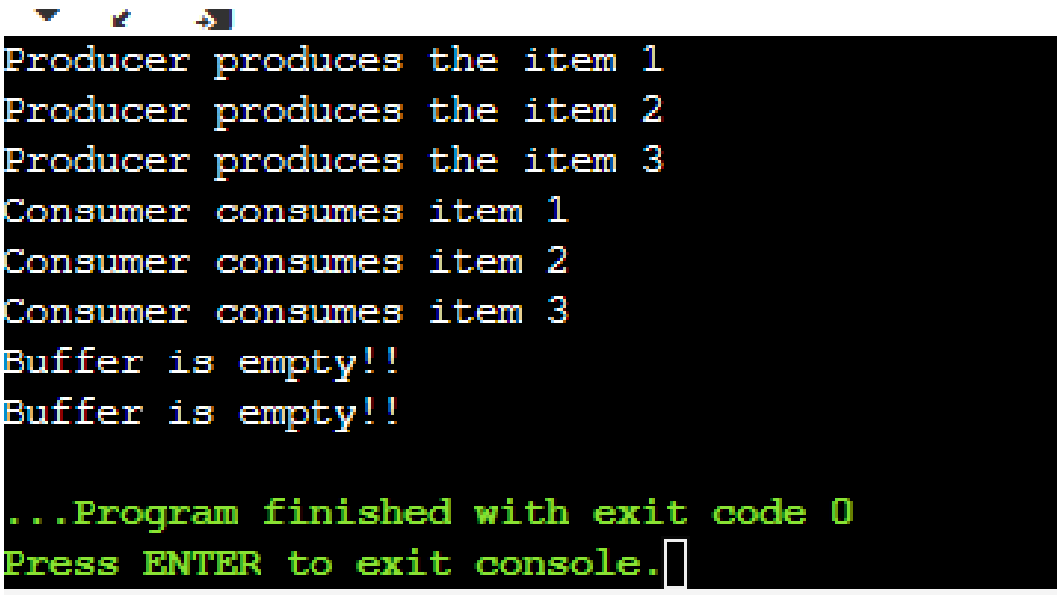
E21CSEU0760


main.cpp

```cpp
#include <iostream>
#include <semaphore.h>
#include <pthread.h>
#include <unistd.h>
using namespace std;

// buffer size
#define BUFFER_SIZE 5

// the buffer
int buffer[BUFFER_SIZE];

// indices for inserting and removing items from the buffer
int in = 0;
int out = 0;

// semaphores
sem_t empty;
sem_t full;  // counts the number of full slots in the buffer
sem_t mutex; // ensures mutual exclusion when accessing the buffer

// producer function
void* producer(void* arg) {
    int item;
    for (int i = 0; i < 3; i++) {
        // produce item
        item = i + 1;

        // wait for empty slot
        sem_wait(&empty);

        // acquire mutex
        sem_wait(&mutex);

        // insert item into buffer
        buffer[in] = item;
        cout << "Producer produces the item " << item << endl;
        in = (in + 1) % BUFFER_SIZE;

        // release mutex
        sem_post(&mutex);

        // signal that a slot is full
```

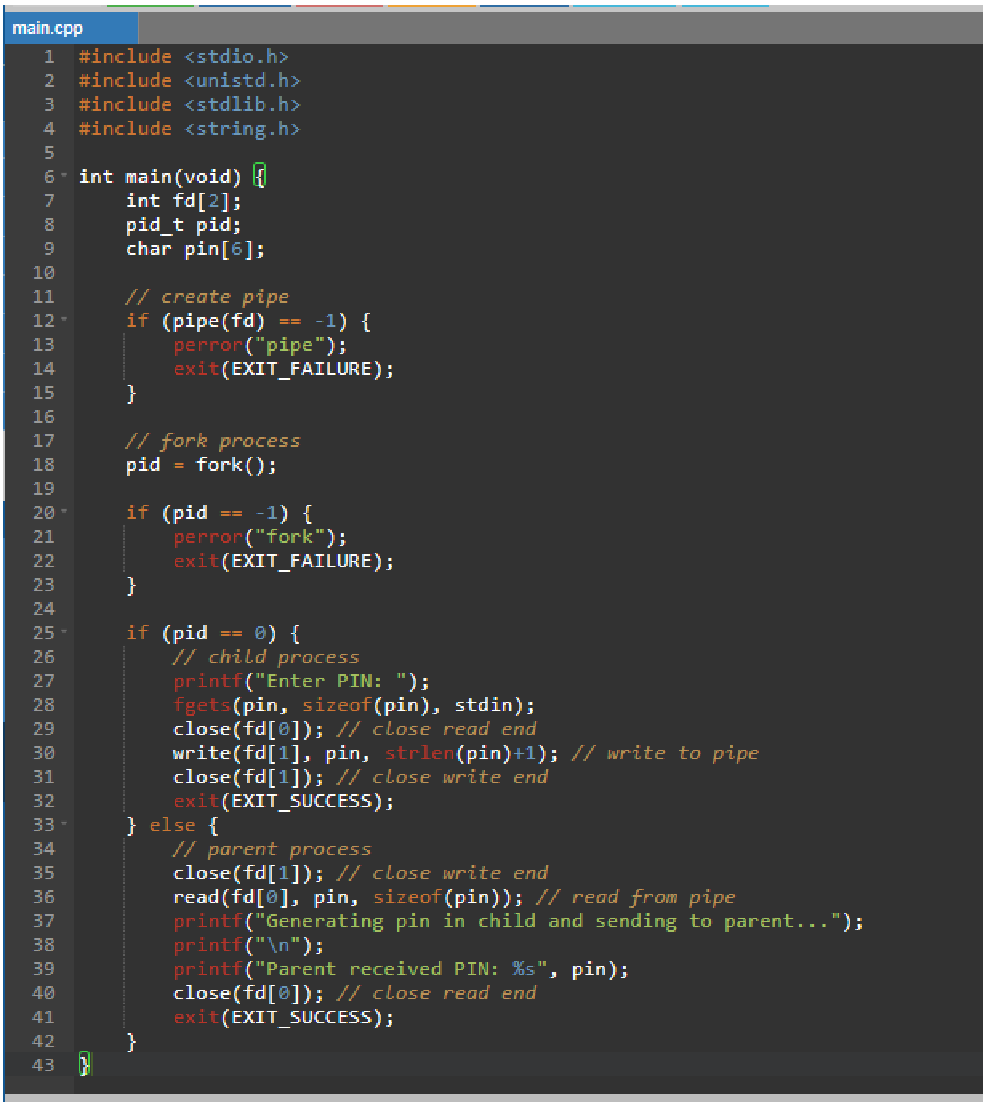


```cpp

        // sleep for random amount of time before producing next item
        usleep(rand() % 1000000);
    }
    pthread_exit(NULL);
}

// consumer function
void* consumer(void* arg) {
    int item;
    for (int i = 0; i < 3; i++) {
        // wait for full slot
        sem_wait(&full);

        // acquire mutex
        sem_wait(&mutex);

        // remove item from buffer
        item = buffer[out];
        cout << "Consumer consumes item " << item << endl;
        out = (out + 1) % BUFFER_SIZE;

        // release mutex
        sem_post(&mutex);

        // signal that a slot is empty
        sem_post(&empty);

        // sleep for random amount of time before consuming next item
        usleep(rand() % 1000000);
    }
    pthread_exit(NULL);
}

int main() {
    // initialize semaphores
    sem_init(&empty, 0, BUFFER_SIZE);
    sem_init(&full, 0, 0);
    sem_init(&mutex, 0, 1);

    // create producer and consumer threads
    pthread_t producerThread, consumerThread;
    pthread_create(&producerThread, NULL, producer, NULL);
    pthread_join(producerThread, NULL);
    pthread_create(&consumerThread, NULL, consumer, NULL);
    pthread_join(consumerThread, NULL);
    cout<<"Buffer is empty!!"<<endl;
    cout<<"Buffer is empty!!";

    // destroy semaphores
    sem_destroy(&empty);
    sem_destroy(&full);
    sem_destroy(&mutex);

    return 0;
}
```

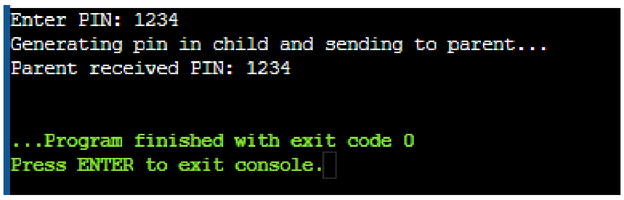


```
Producer produces the item 1
Producer produces the item 2
Producer produces the item 3
Consumer consumes item 1
Consumer consumes item 2
Consumer consumes item 3
Buffer is empty!!
Buffer is empty!!

...Program finished with exit code 0
Press ENTER to exit console.
```

TASK 2



```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>

int main(void) {
    int fd[2];
    pid_t pid;
    char pin[6];

    // create pipe
    if (pipe(fd) == -1) {
        perror("pipe");
        exit(EXIT_FAILURE);
    }

    // fork process
    pid = fork();

    if (pid == -1) {
        perror("fork");
        exit(EXIT_FAILURE);
    }

    if (pid == 0) {
        // child process
        printf("Enter PIN: ");
        fgets(pin, sizeof(pin), stdin);
        close(fd[0]); // close read end
        write(fd[1], pin, strlen(pin)+1); // write to pipe
        close(fd[1]); // close write end
        exit(EXIT_SUCCESS);
    } else {
        // parent process
        close(fd[1]); // close write end
        read(fd[0], pin, sizeof(pin)); // read from pipe
        printf("Generating pin in child and sending to parent...");
        printf("\n");
        printf("Parent received PIN: %s", pin);
        close(fd[0]); // close read end
        exit(EXIT_SUCCESS);
    }
}
```

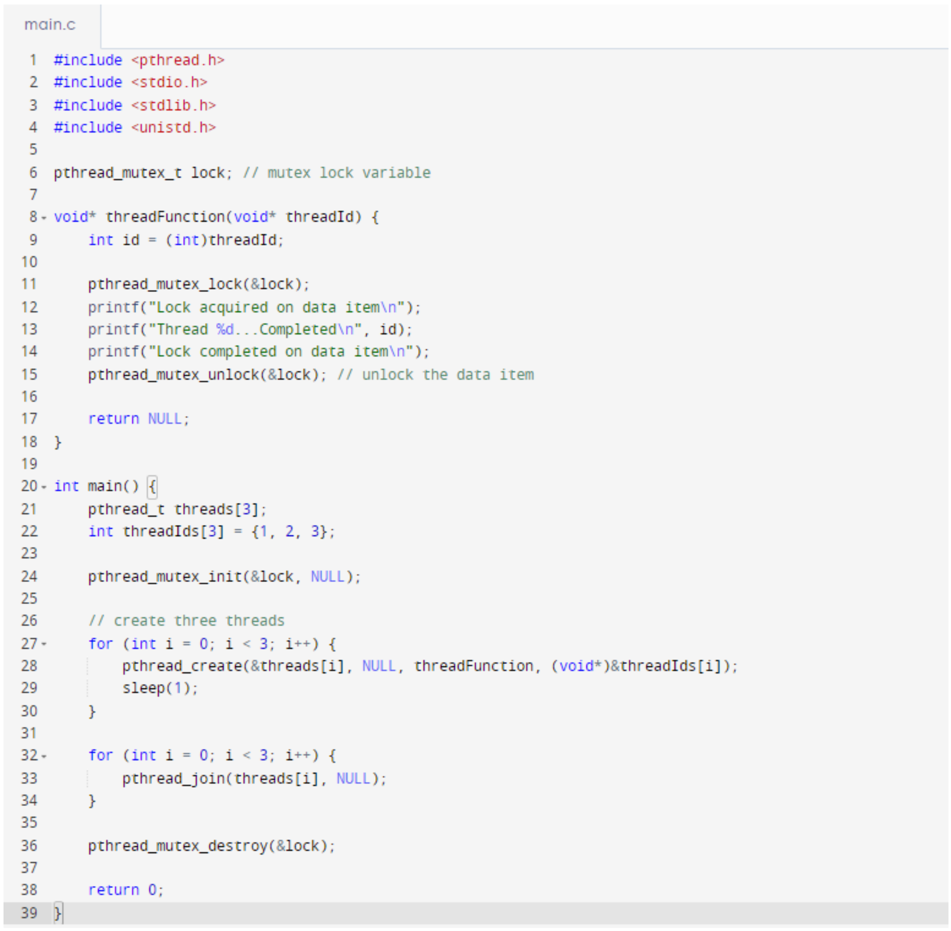


```
Enter PIN: 1234
Generating pin in child and sending to parent...
Parent received PIN: 1234


...Program finished with exit code 0
Press ENTER to exit console.
```

Soumya Dubey

E21CSEU0760

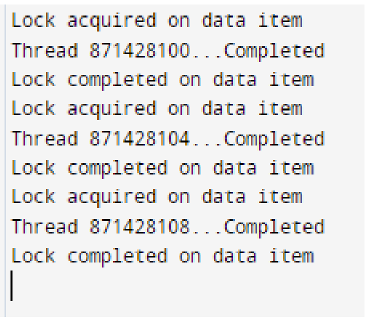
Task 1:



main.c

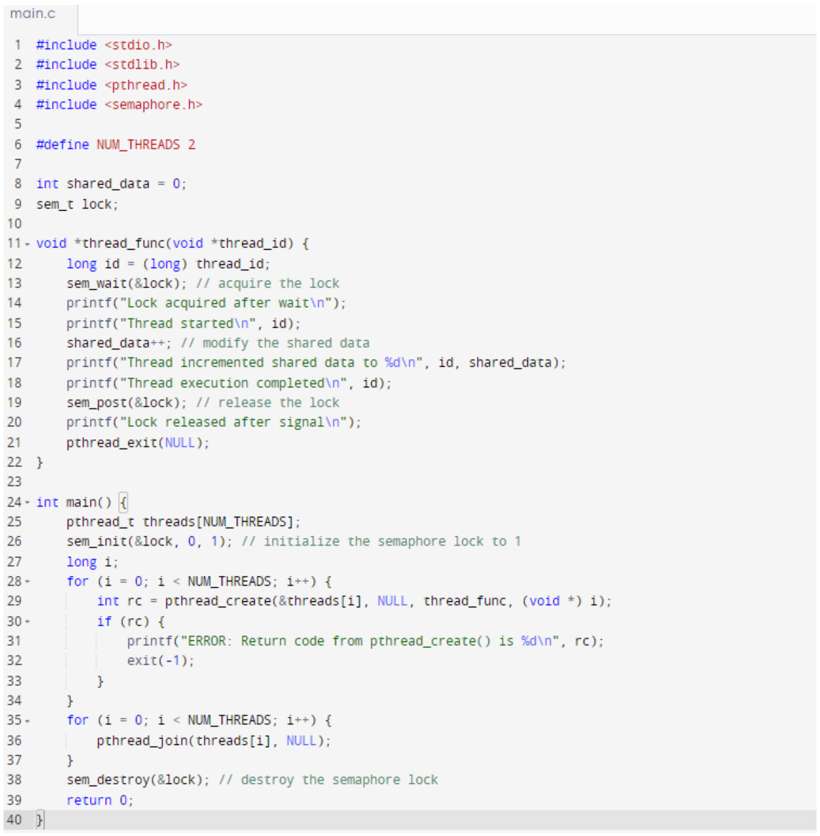
```c
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

pthread_mutex_t lock; // mutex lock variable

void* threadFunction(void* threadId) {
    int id = (int)threadId;

    pthread_mutex_lock(&lock);
    printf("Lock acquired on data item\n");
    printf("Thread %d...Completed\n", id);
    printf("Lock completed on data item\n");
    pthread_mutex_unlock(&lock); // unlock the data item

    return NULL;
}

int main() {
    pthread_t threads[3];
    int threadIds[3] = {1, 2, 3};

    pthread_mutex_init(&lock, NULL);

    // create three threads
    for (int i = 0; i < 3; i++) {
        pthread_create(&threads[i], NULL, threadFunction, (void*)&threadIds[i]);
        sleep(1);
    }

    for (int i = 0; i < 3; i++) {
        pthread_join(threads[i], NULL);
    }

    pthread_mutex_destroy(&lock);

    return 0;
}
```

Output:



```
Lock acquired on data item
Thread 871428100...Completed
Lock completed on data item
Lock acquired on data item
Thread 871428104...Completed
Lock completed on data item
Lock acquired on data item
Thread 871428108...Completed
Lock completed on data item
```

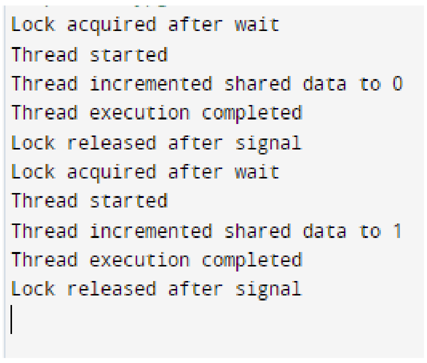
Task 2:



main.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <semaphore.h>

#define NUM_THREADS 2

int shared_data = 0;
sem_t lock;

void *thread_func(void *thread_id) {
    long id = (long) thread_id;
    sem_wait(&lock); // acquire the lock
    printf("Lock acquired after wait\n");
    printf("Thread started\n", id);
    shared_data++; // modify the shared data
    printf("Thread incremented shared data to %d\n", id, shared_data);
    printf("Thread execution completed\n", id);
    sem_post(&lock); // release the lock
    printf("Lock released after signal\n");
    pthread_exit(NULL);
}

int main() {
    pthread_t threads[NUM_THREADS];
    sem_init(&lock, 0, 1); // initialize the semaphore lock to 1
    long i;
    for (i = 0; i < NUM_THREADS; i++) {
        int rc = pthread_create(&threads[i], NULL, thread_func, (void *) i);
        if (rc) {
            printf("ERROR: Return code from pthread_create() is %d\n", rc);
            exit(-1);
        }
    }
    for (i = 0; i < NUM_THREADS; i++) {
        pthread_join(threads[i], NULL);
    }
    sem_destroy(&lock); // destroy the semaphore lock
    return 0;
}
```

Output:

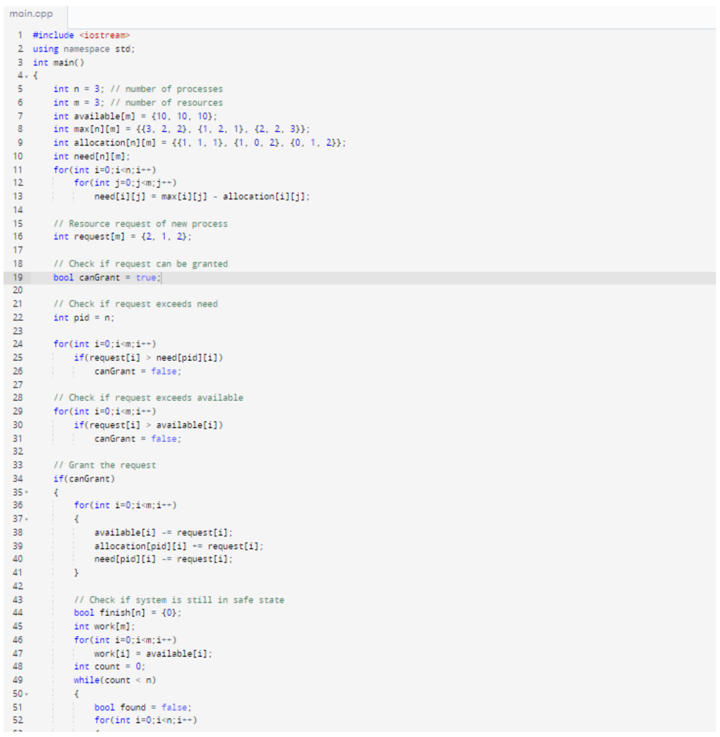


```
Lock acquired after wait
Thread started
Thread incremented shared data to 0
Thread execution completed
Lock released after signal
Lock acquired after wait
Thread started
Thread incremented shared data to 1
Thread execution completed
Lock released after signal
```

Lab 09

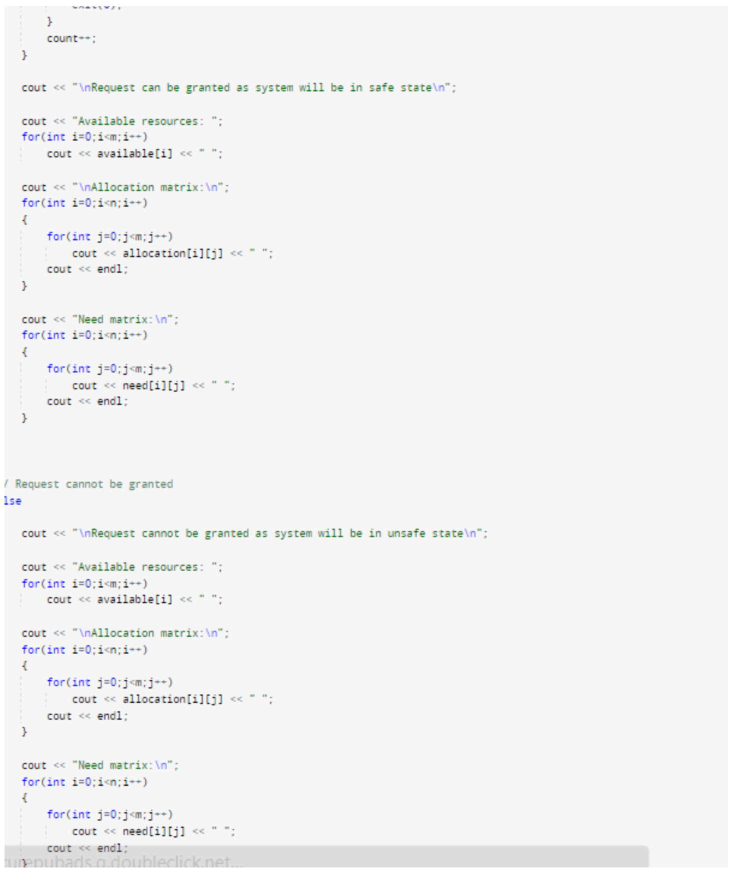
Soumya Dubey

E21CSEU0760



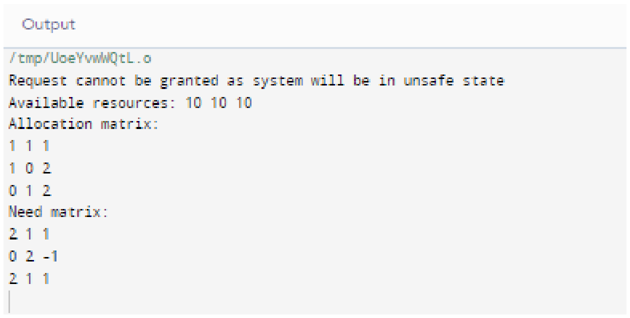
main.cpp

```cpp
#include <iostream>
using namespace std;
int main()
{
    int n = 3; // number of processes
    int m = 3; // number of resources
    int available[m] = {10, 10, 10};
    int max[n][m] = {{3, 2, 2}, {1, 2, 1}, {2, 2, 3}};
    int allocation[n][m] = {{1, 1, 1}, {1, 0, 2}, {0, 1, 2}};
    int need[n][m];
    for(int i=0;i<n;i++)
        for(int j=0;j<m;j++)
            need[i][j] = max[i][j] - allocation[i][j];

    // Resource request of new process
    int request[m] = {2, 1, 2};

    // Check if request can be granted
    bool canGrant = true;

    // Check if request exceeds need
    int pid = n;

    for(int i=0;i<m;i++)
        if(request[i] > need[pid][i])
            canGrant = false;

    // Check if request exceeds available
    for(int i=0;i<m;i++)
        if(request[i] > available[i])
            canGrant = false;

    // Grant the request
    if(canGrant)
    {
        for(int i=0;i<m;i++)
        {
            available[i] -= request[i];
            allocation[pid][i] += request[i];
            need[pid][i] -= request[i];
        }

        // Check if system is still in safe state
        bool finish[n] = {0};
        int work[m];
        for(int i=0;i<m;i++)
            work[i] = available[i];
        int count = 0;
        while(count < n)
        {
            bool found = false;
            for(int i=0;i<n;i++)
```



```
            exit(0);
        }
        count++;
    }

    cout << "\nRequest can be granted as system will be in safe state\n";

    cout << "Available resources: ";
    for(int i=0;i<m;i++)
        cout << available[i] << " ";

    cout << "\nAllocation matrix:\n";
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<m;j++)
            cout << allocation[i][j] << " ";
        cout << endl;
    }

    cout << "Need matrix:\n";
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<m;j++)
            cout << need[i][j] << " ";
        cout << endl;
    }



/ Request cannot be granted
lse

    cout << "\nRequest cannot be granted as system will be in unsafe state\n";

    cout << "Available resources: ";
    for(int i=0;i<m;i++)
        cout << available[i] << " ";

    cout << "\nAllocation matrix:\n";
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<m;j++)
            cout << allocation[i][j] << " ";
        cout << endl;
    }

    cout << "Need matrix:\n";
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<m;j++)
            cout << need[i][j] << " ";
        cout << endl;
    }
```

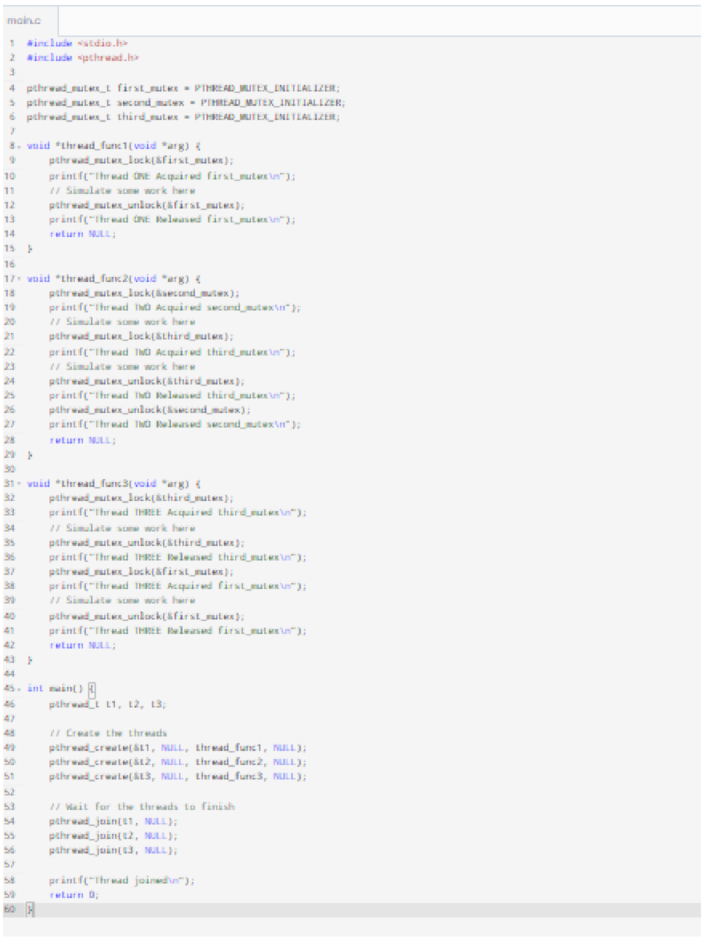
tpubads.g.doubleclick.net...



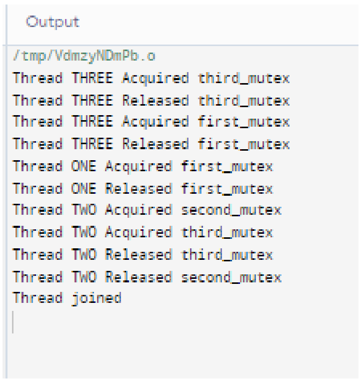
Output

```
/tmp/UoeYvwWQtL.o
Request cannot be granted as system will be in unsafe state
Available resources: 10 10 10
Allocation matrix:
1 1 1
1 0 2
0 1 2
Need matrix:
2 1 1
0 2 -1
2 1 1
```

Task 2


main.c

```c
#include <stdio.h>
#include <pthread.h>

pthread_mutex_t first_mutex = PTHREAD_MUTEX_INITIALIZER;
pthread_mutex_t second_mutex = PTHREAD_MUTEX_INITIALIZER;
pthread_mutex_t third_mutex = PTHREAD_MUTEX_INITIALIZER;

void *thread_func1(void *arg) {
    pthread_mutex_lock(&first_mutex);
    printf("Thread ONE Acquired first_mutex\n");
    // Simulate some work here
    pthread_mutex_unlock(&first_mutex);
    printf("Thread ONE Released first_mutex\n");
    return NULL;
}

void *thread_func2(void *arg) {
    pthread_mutex_lock(&second_mutex);
    printf("Thread TWO Acquired second_mutex\n");
    // Simulate some work here
    pthread_mutex_lock(&third_mutex);
    printf("Thread TWO Acquired third_mutex\n");
    // Simulate some work here
    pthread_mutex_unlock(&third_mutex);
    printf("Thread TWO Released third_mutex\n");
    pthread_mutex_unlock(&second_mutex);
    printf("Thread TWO Released second_mutex\n");
    return NULL;
}

void *thread_func3(void *arg) {
    pthread_mutex_lock(&third_mutex);
    printf("Thread THREE Acquired third_mutex\n");
    // Simulate some work here
    pthread_mutex_unlock(&third_mutex);
    printf("Thread THREE Released third_mutex\n");
    pthread_mutex_lock(&first_mutex);
    printf("Thread THREE Acquired first_mutex\n");
    // Simulate some work here
    pthread_mutex_unlock(&first_mutex);
    printf("Thread THREE Released first_mutex\n");
    return NULL;
}

int main() {
    pthread_t t1, t2, t3;

    // Create the threads
    pthread_create(&t1, NULL, thread_func1, NULL);
    pthread_create(&t2, NULL, thread_func2, NULL);
    pthread_create(&t3, NULL, thread_func3, NULL);

    // Wait for the threads to finish
    pthread_join(t1, NULL);
    pthread_join(t2, NULL);
    pthread_join(t3, NULL);

    printf("Thread joined\n");
    return 0;
}
```


Output

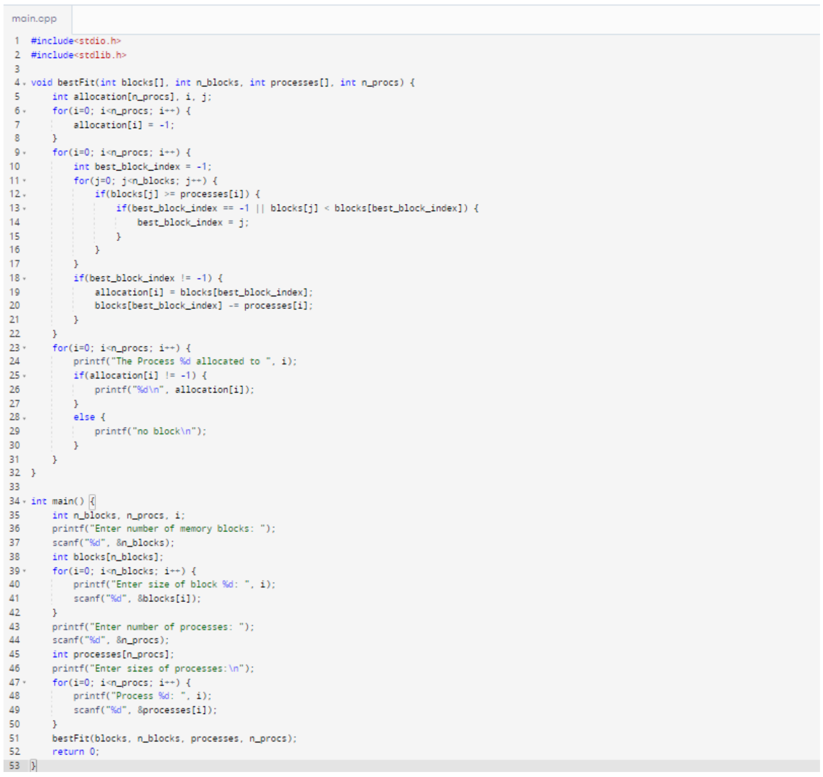
```
/tmp/VdmzyNDmPb.o
Thread THREE Acquired third_mutex
Thread THREE Released third_mutex
Thread THREE Acquired first_mutex
Thread THREE Released first_mutex
Thread ONE Acquired first_mutex
Thread ONE Released first_mutex
Thread TWO Acquired second_mutex
Thread TWO Acquired third_mutex
Thread TWO Released third_mutex
Thread TWO Released second_mutex
Thread joined
```

Lab 10

Soumya Dubey

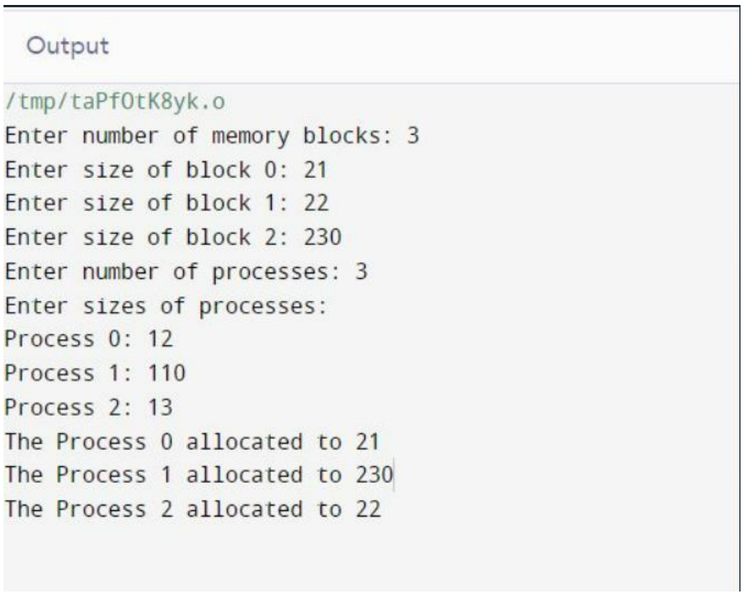
E21CSEU0760

Task 1



main.cpp

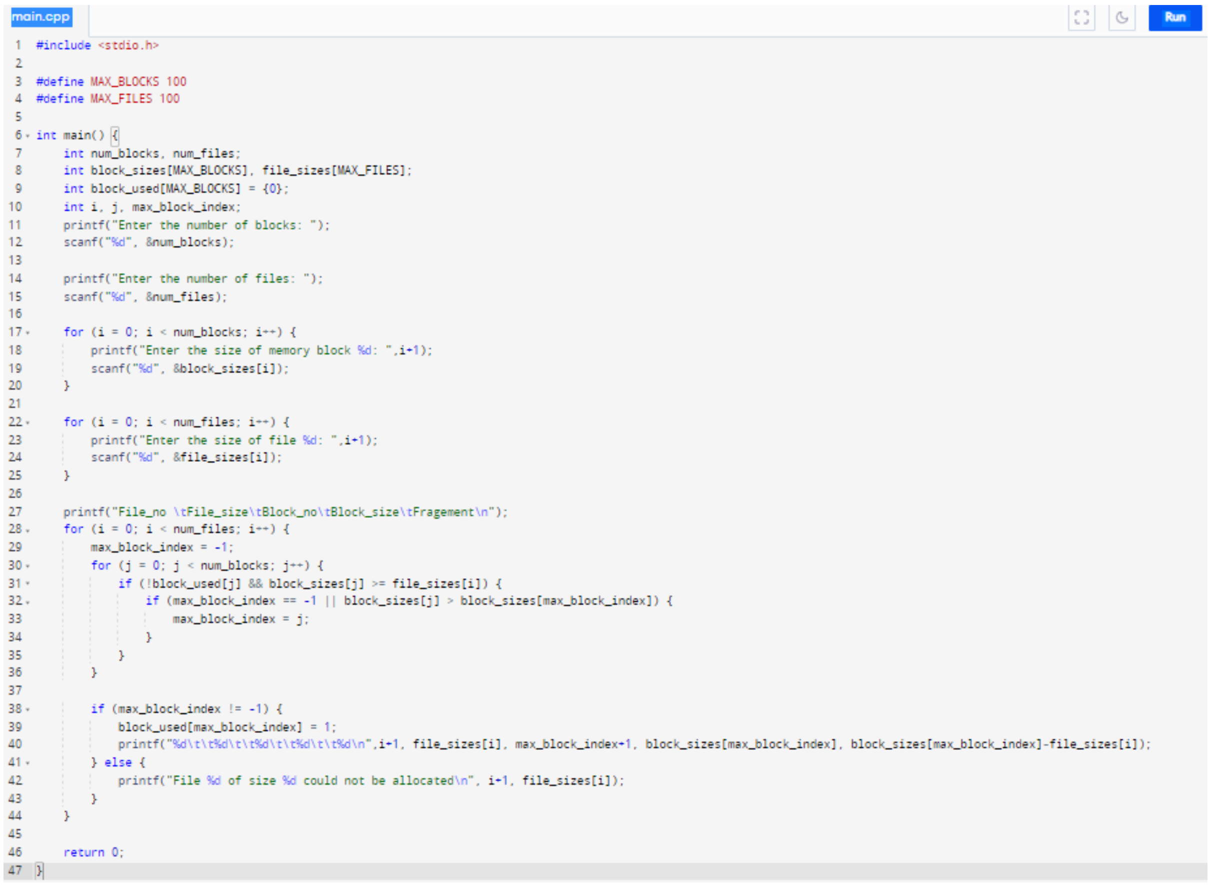
```c
#include<stdio.h>
#include<stdlib.h>

void bestFit(int blocks[], int n_blocks, int processes[], int n_procs) {
    int allocation[n_procs], i, j;
    for(i=0; i<n_procs; i++) {
        allocation[i] = -1;
    }
    for(i=0; i<n_procs; i++) {
        int best_block_index = -1;
        for(j=0; j<n_blocks; j++) {
            if(blocks[j] >= processes[i]) {
                if(best_block_index == -1 || blocks[j] < blocks[best_block_index]) {
                    best_block_index = j;
                }
            }
        }
        if(best_block_index != -1) {
            allocation[i] = blocks[best_block_index];
            blocks[best_block_index] -= processes[i];
        }
    }
    for(i=0; i<n_procs; i++) {
        printf("The Process %d allocated to ", i);
        if(allocation[i] != -1) {
            printf("%d\n", allocation[i]);
        }
        else {
            printf("no block\n");
        }
    }
}

int main() {
    int n_blocks, n_procs, i;
    printf("Enter number of memory blocks: ");
    scanf("%d", &n_blocks);
    int blocks[n_blocks];
    for(i=0; i<n_blocks; i++) {
        printf("Enter size of block %d: ", i);
        scanf("%d", &blocks[i]);
    }
    printf("Enter number of processes: ");
    scanf("%d", &n_procs);
    int processes[n_procs];
    printf("Enter sizes of processes:\n");
    for(i=0; i<n_procs; i++) {
        printf("Process %d: ", i);
        scanf("%d", &processes[i]);
    }
    bestFit(blocks, n_blocks, processes, n_procs);
    return 0;
}
```



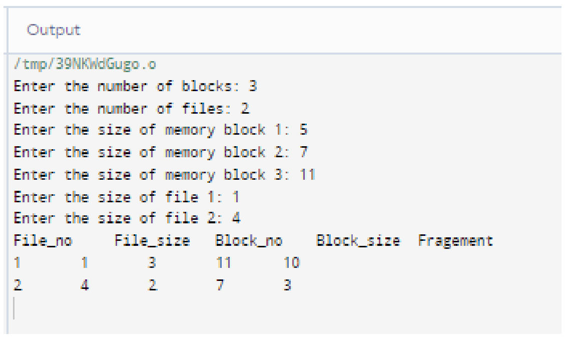
Output

```
/tmp/taPfOtK8yk.o
Enter number of memory blocks: 3
Enter size of block 0: 21
Enter size of block 1: 22
Enter size of block 2: 230
Enter number of processes: 3
Enter sizes of processes:
Process 0: 12
Process 1: 110
Process 2: 13
The Process 0 allocated to 21
The Process 1 allocated to 230
The Process 2 allocated to 22
```

Task 2



```c
#include <stdio.h>

#define MAX_BLOCKS 100
#define MAX_FILES 100

int main() {
    int num_blocks, num_files;
    int block_sizes[MAX_BLOCKS], file_sizes[MAX_FILES];
    int block_used[MAX_BLOCKS] = {0};
    int i, j, max_block_index;
    printf("Enter the number of blocks: ");
    scanf("%d", &num_blocks);

    printf("Enter the number of files: ");
    scanf("%d", &num_files);

    for (i = 0; i < num_blocks; i++) {
        printf("Enter the size of memory block %d: ",i+1);
        scanf("%d", &block_sizes[i]);
    }

    for (i = 0; i < num_files; i++) {
        printf("Enter the size of file %d: ",i+1);
        scanf("%d", &file_sizes[i]);
    }

    printf("File_no \tFile_size\tBlock_no\tBlock_size\tFragement\n");
    for (i = 0; i < num_files; i++) {
        max_block_index = -1;
        for (j = 0; j < num_blocks; j++) {
            if (!block_used[j] && block_sizes[j] >= file_sizes[i]) {
                if (max_block_index == -1 || block_sizes[j] > block_sizes[max_block_index]) {
                    max_block_index = j;
                }
            }
        }

        if (max_block_index != -1) {
            block_used[max_block_index] = 1;
            printf("%d\t\t%d\t\t%d\t\t%d\t\t%d\n",i+1, file_sizes[i], max_block_index+1, block_sizes[max_block_index], block_sizes[max_block_index]-file_sizes[i]);
        } else {
            printf("File %d of size %d could not be allocated\n", i+1, file_sizes[i]);
        }
    }

    return 0;
}
```



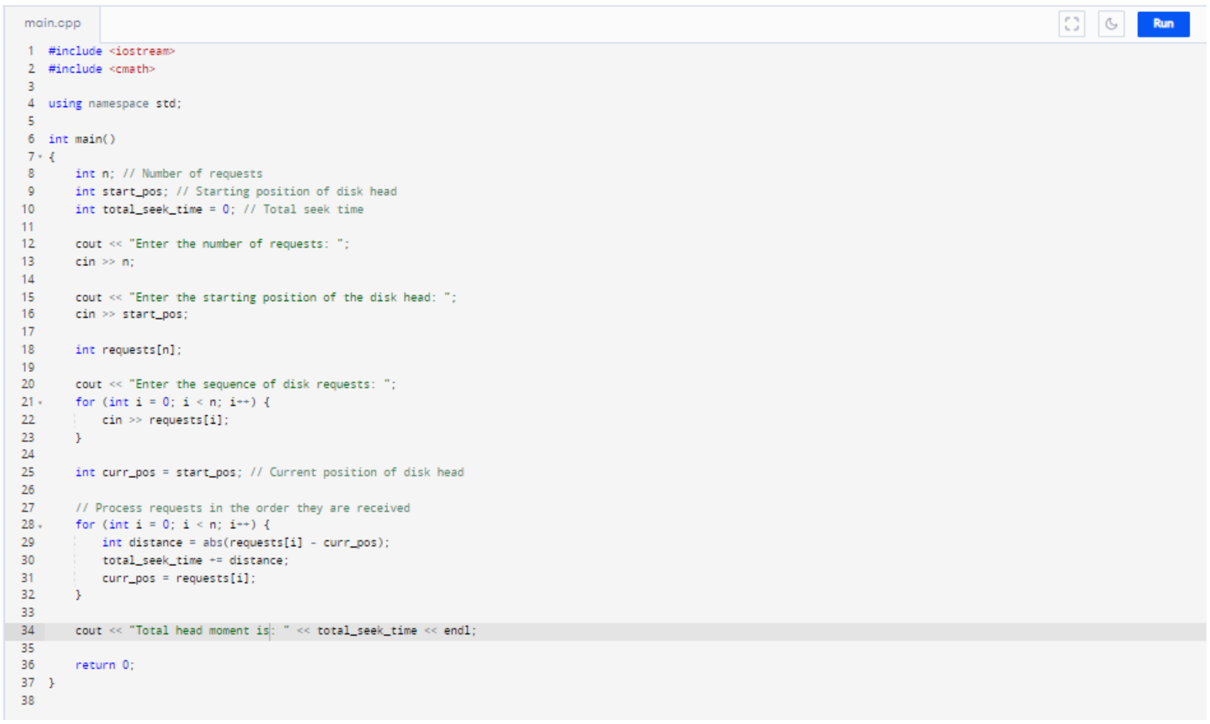
```
/tmp/39NKWdGugo.o
Enter the number of blocks: 3
Enter the number of files: 2
Enter the size of memory block 1: 5
Enter the size of memory block 2: 7
Enter the size of memory block 3: 11
Enter the size of file 1: 1
Enter the size of file 2: 4
File_no     File_size    Block_no    Block_size   Fragement
1        1        3        11       10
2        4        2        7        3
```

Lab 11

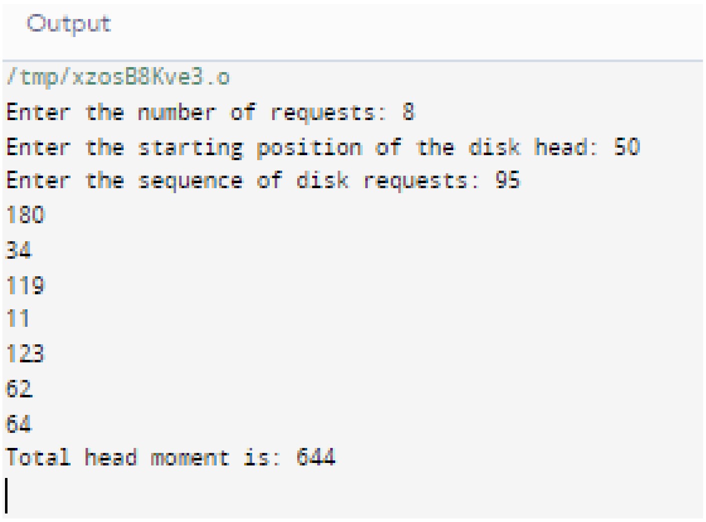
Soumya Dubey

E21CSEU0760

Task 1

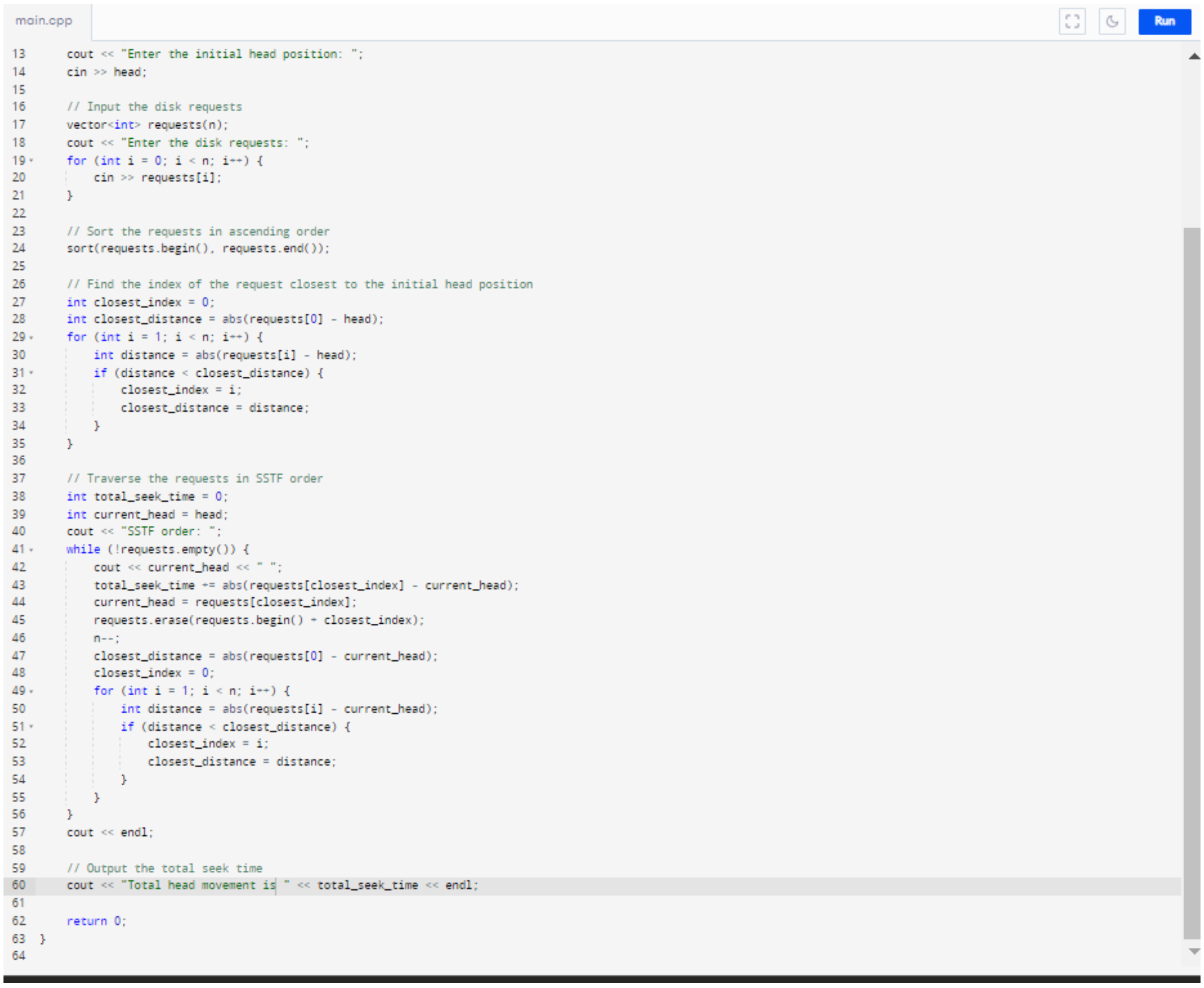


```cpp
#include <iostream>
#include <cmath>

using namespace std;

int main()
{
    int n; // Number of requests
    int start_pos; // Starting position of disk head
    int total_seek_time = 0; // Total seek time

    cout << "Enter the number of requests: ";
    cin >> n;

    cout << "Enter the starting position of the disk head: ";
    cin >> start_pos;

    int requests[n];

    cout << "Enter the sequence of disk requests: ";
    for (int i = 0; i < n; i++) {
        cin >> requests[i];
    }

    int curr_pos = start_pos; // Current position of disk head

    // Process requests in the order they are received
    for (int i = 0; i < n; i++) {
        int distance = abs(requests[i] - curr_pos);
        total_seek_time += distance;
        curr_pos = requests[i];
    }

    cout << "Total head moment is: " << total_seek_time << endl;

    return 0;
}
```

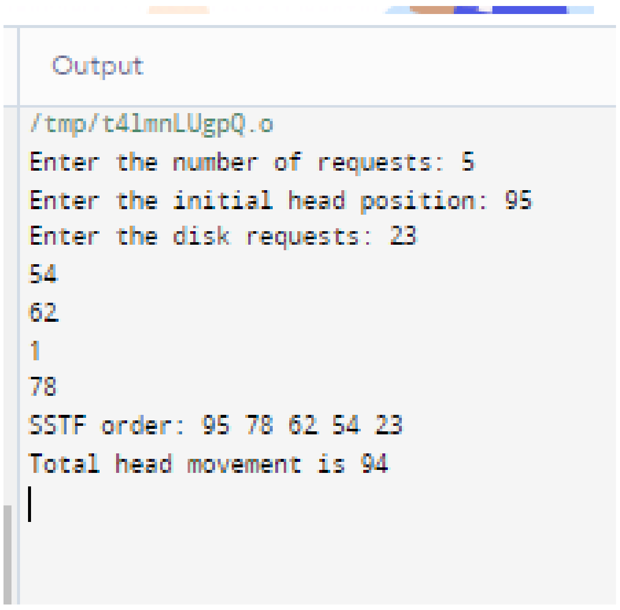


```
/tmp/xzosB8Kve3.o
Enter the number of requests: 8
Enter the starting position of the disk head: 50
Enter the sequence of disk requests: 95
180
34
119
11
123
62
64
Total head moment is: 644
```

Task 2



```cpp
    cout << "Enter the initial head position: ";
    cin >> head;

    // Input the disk requests
    vector<int> requests(n);
    cout << "Enter the disk requests: ";
    for (int i = 0; i < n; i++) {
        cin >> requests[i];
    }

    // Sort the requests in ascending order
    sort(requests.begin(), requests.end());

    // Find the index of the request closest to the initial head position
    int closest_index = 0;
    int closest_distance = abs(requests[0] - head);
    for (int i = 1; i < n; i++) {
        int distance = abs(requests[i] - head);
        if (distance < closest_distance) {
            closest_index = i;
            closest_distance = distance;
        }
    }

    // Traverse the requests in SSTF order
    int total_seek_time = 0;
    int current_head = head;
    cout << "SSTF order: ";
    while (!requests.empty()) {
        cout << current_head << " ";
        total_seek_time += abs(requests[closest_index] - current_head);
        current_head = requests[closest_index];
        requests.erase(requests.begin() + closest_index);
        n--;
        closest_distance = abs(requests[0] - current_head);
        closest_index = 0;
        for (int i = 1; i < n; i++) {
            int distance = abs(requests[i] - current_head);
            if (distance < closest_distance) {
                closest_index = i;
                closest_distance = distance;
            }
        }
    }
    cout << endl;

    // Output the total seek time
    cout << "Total head movement is " << total_seek_time << endl;

    return 0;
}
```



```
/tmp/t4lmnLUgpQ.o
Enter the number of requests: 5
Enter the initial head position: 95
Enter the disk requests: 23
54
62
1
78
SSTF order: 95 78 62 54 23
Total head movement is 94
```

+