P.E.S COLLEGE OF ENGINEERING,
MANDYA,571401

(An Autonomous & Govt. Aided Institution, Affilitated to VTU, Belagavi)

Assignment Report
on

"UNIX SYSTEM PROGRAMMING"

In partial fulfilment of the requirement
for the award of the Degree

Bachelor of Engineering
In

COMPUTER SCIENCE AND ENGINEERING

Submitted by

MITHUN RP [4PS21CS055]
MANOJGOWDA KS [4PS21CS049]
MADHAV ADITHYA M S [4PS21CS055]
MITHUN DEV M [4PS21CS054]

Under the guidance of

Mrs. Deepika
Assistant Professor, Dept. of CS&E,
P.E.S.C.E, Mandya.

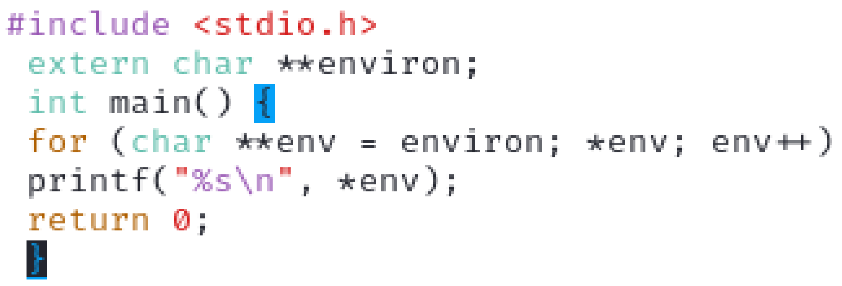
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
P.E.S. College of Engineering, Mandya,
2023-2024

CONTENT

PROGRAMMING ASSIGNMENT

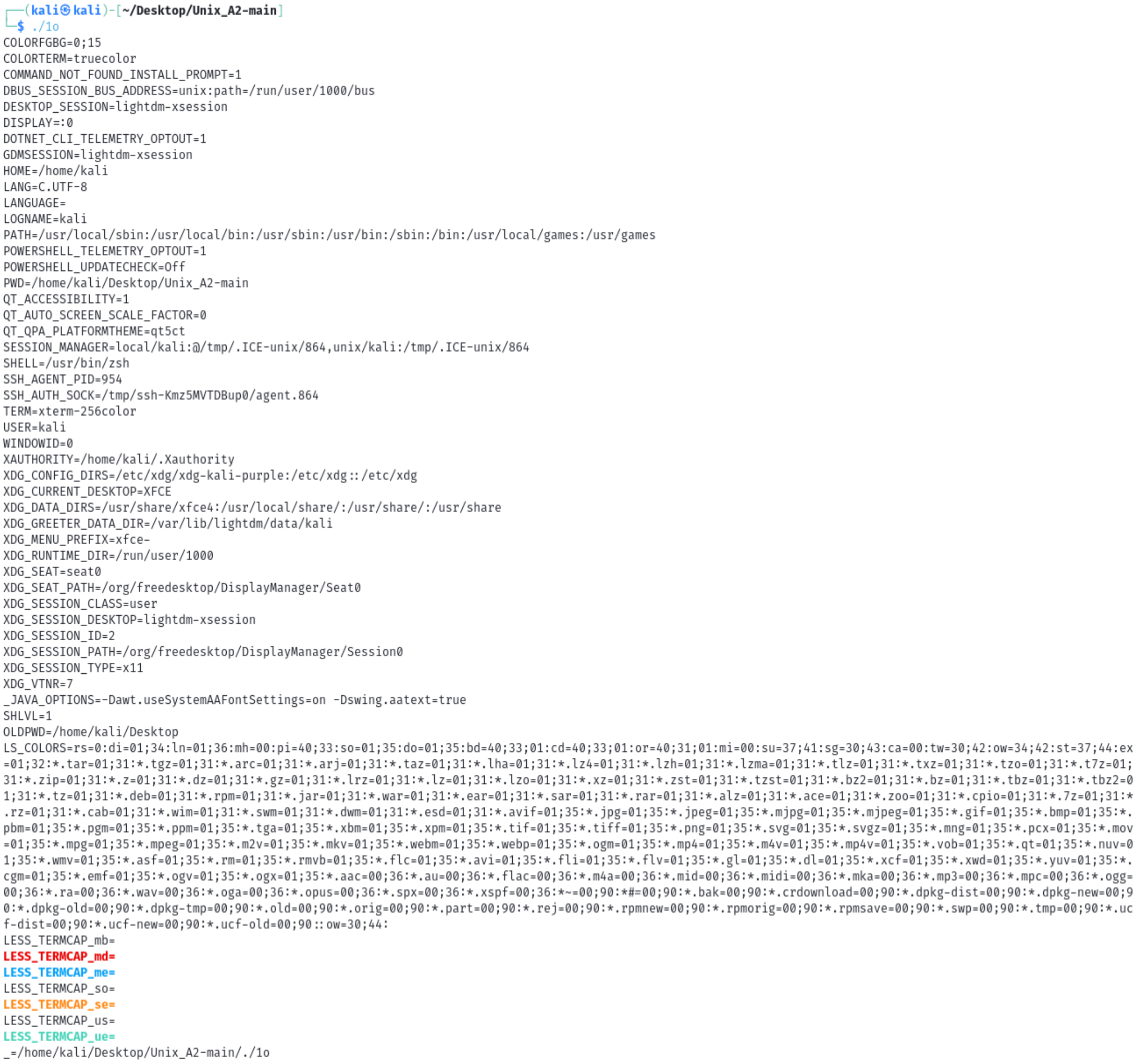
1.) .Write a C/C++ program to that outputs the contents of its environment list.

Code:



```
#include <stdio.h>
 extern char **environ;
 int main() {
 for (char **env = environ; *env; env++)
 printf("%s\n", *env);
 return 0;
 }
```

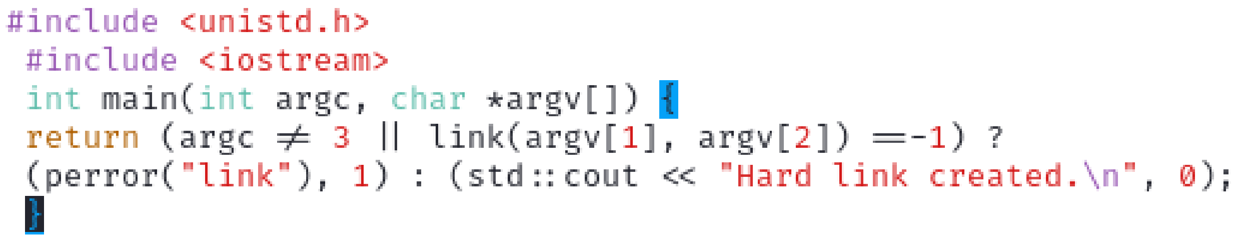
Output:



```
┌──(kali㉿kali)-[~/Desktop/Unix_A2-main]
└─$ ./1o
COLORFGBG=0;15
COLORTERM=truecolor
COMMAND_NOT_FOUND_INSTALL_PROMPT=1
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
DESKTOP_SESSION=lightdm-xsession
DISPLAY=:0
DOTNET_CLI_TELEMETRY_OPTOUT=1
GDMSESSION=lightdm-xsession
HOME=/home/kali
LANG=C.UTF-8
LANGUAGE=
LOGNAME=kali
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/local/games:/usr/games
POWERSHELL_TELEMETRY_OPTOUT=1
POWERSHELL_UPDATECHECK=Off
PWD=/home/kali/Desktop/Unix_A2-main
QT_ACCESSIBILITY=1
QT_AUTO_SCREEN_SCALE_FACTOR=0
QT_QPA_PLATFORMTHEME=qt5ct
SESSION_MANAGER=local/kali:@/tmp/.ICE-unix/864,unix/kali:/tmp/.ICE-unix/864
SHELL=/usr/bin/zsh
SSH_AGENT_PID=954
SSH_AUTH_SOCK=/tmp/ssh-Kmz5MVTDBup0/agent.864
TERM=xterm-256color
USER=kali
WINDOWID=0
XAUTHORITY=/home/kali/.Xauthority
XDG_CONFIG_DIRS=/etc/xdg/xdg-kali-purple:/etc/xdg::/etc/xdg
XDG_CURRENT_DESKTOP=XFCE
XDG_DATA_DIRS=/usr/share/xfce4:/usr/local/share/:/usr/share/:/usr/share
XDG_GREETER_DATA_DIR=/var/lib/lightdm/data/kali
XDG_MENU_PREFIX=xfce-
XDG_RUNTIME_DIR=/run/user/1000
XDG_SEAT=seat0
XDG_SEAT_PATH=/org/freedesktop/DisplayManager/Seat0
XDG_SESSION_CLASS=user
XDG_SESSION_DESKTOP=lightdm-xsession
XDG_SESSION_ID=2
XDG_SESSION_PATH=/org/freedesktop/DisplayManager/Session0
XDG_SESSION_TYPE=x11
XDG_VTNR=7
_JAVA_OPTIONS=-Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
SHLVL=1
OLDPWD=/home/kali/Desktop
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31;01:mi=00:su=37;41:sg=30;43:ca=00:tw=30;42:ow=34;42:st=37;44:ex
=01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:*.t7z=01;
31:*.zip=01;31:*.z=01;31:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.lz=01;31:*.lzo=01;31:*.xz=01;31:*.zst=01;31:*.tzst=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.tbz2=0
1;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;31:*.sar=01;31:*.rar=01;31:*.alz=01;31:*.ace=01;31:*.zoo=01;31:*.cpio=01;31:*.7z=01;31:*
.rz=01;31:*.cab=01;31:*.wim=01;31:*.swm=01;31:*.dwm=01;31:*.esd=01;31:*.avif=01;35:*.jpg=01;35:*.jpeg=01;35:*.mjpg=01;35:*.mjpeg=01;35:*.gif=01;35:*.bmp=01;35:*.
pbm=01;35:*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm=01;35:*.xpm=01;35:*.tif=01;35:*.tiff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.pcx=01;35:*.mov
=01;35:*.mpg=01;35:*.mpeg=01;35:*.m2v=01;35:*.mkv=01;35:*.webm=01;35:*.webp=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=0
1;35:*.wmv=01;35:*.asf=01;35:*.rm=01;35:*.rmvb=01;35:*.flc=01;35:*.avi=01;35:*.fli=01;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=01;35:*.xwd=01;35:*.yuv=01;35:*.
cgm=01;35:*.emf=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=00;36:*.au=00;36:*.flac=00;36:*.m4a=00;36:*.mid=00;36:*.midi=00;36:*.mka=00;36:*.mp3=00;36:*.mpc=00;36:*.ogg=
00;36:*.ra=00;36:*.wav=00;36:*.oga=00;36:*.opus=00;36:*.spx=00;36:*.xspf=00;36:*~=00;90:*#=00;90:*.bak=00;90:*.crdownload=00;90:*.dpkg-dist=00;90:*.dpkg-new=00;9
0:*.dpkg-old=00;90:*.dpkg-tmp=00;90:*.old=00;90:*.orig=00;90:*.part=00;90:*.rej=00;90:*.rpmnew=00;90:*.rpmorig=00;90:*.rpmsave=00;90:*.swp=00;90:*.tmp=00;90:*.uc
f-dist=00;90:*.ucf-new=00;90:*.ucf-old=00;90::ow=30;44:
LESS_TERMCAP_mb=
LESS_TERMCAP_md=
LESS_TERMCAP_me=
LESS_TERMCAP_so=
LESS_TERMCAP_se=
LESS_TERMCAP_us=
LESS_TERMCAP_ue=
_=/home/kali/Desktop/Unix_A2-main/./1o
```

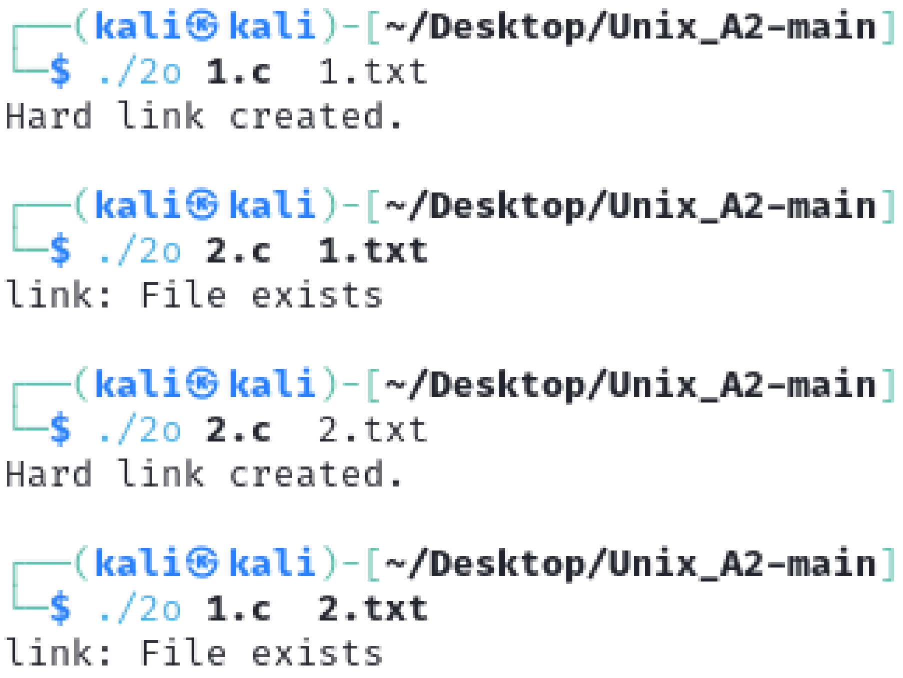
2.) Write a C/C++ program to emulate the Unix ln command.

Code:



```
#include <unistd.h>
 #include <iostream>
 int main(int argc, char *argv[]) {
 return (argc ≠ 3 || link(argv[1], argv[2]) ==-1) ?
 (perror("link"), 1) : (std::cout << "Hard link created.\n", 0);
 }
```

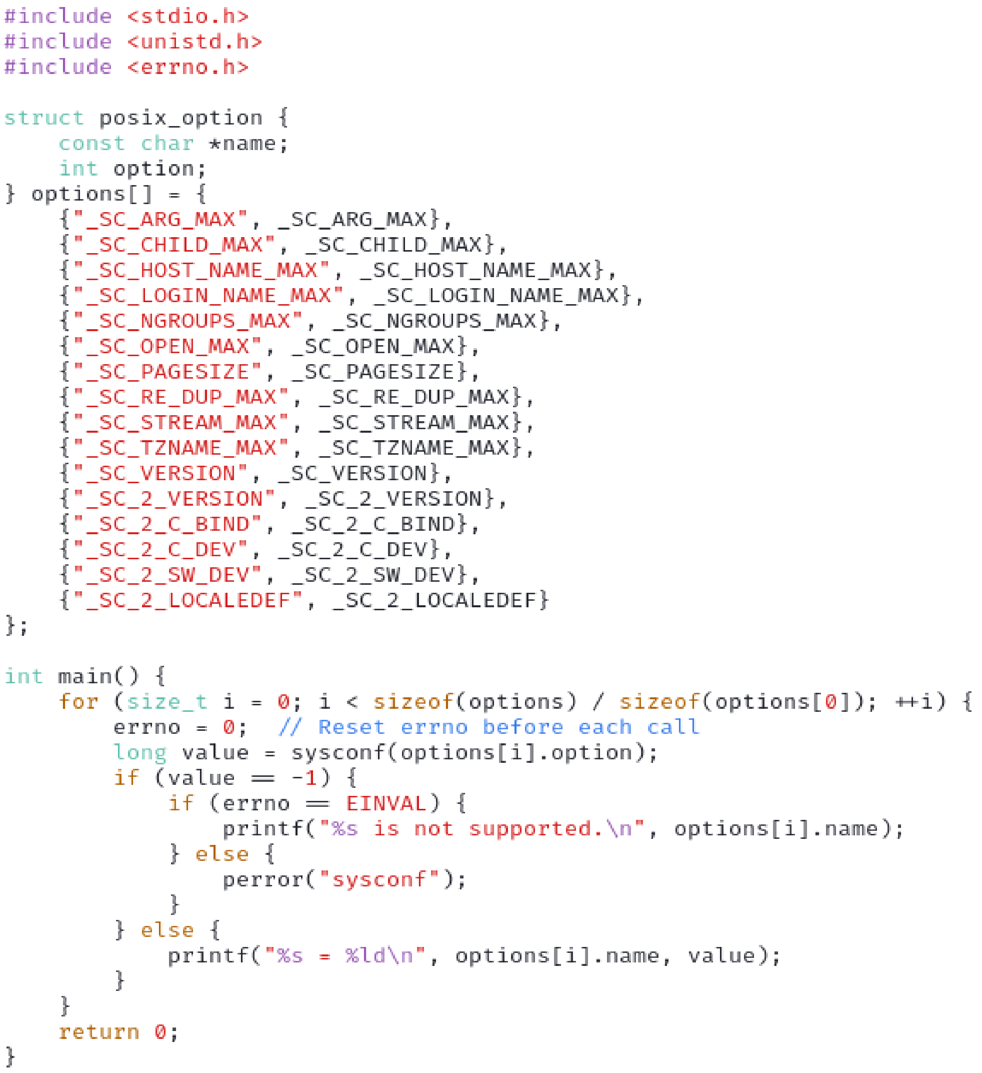
Output:



```
┌──(kali㉿kali)-[~/Desktop/Unix_A2-main]
└─$ ./2o 1.c  1.txt
Hard link created.

┌──(kali㉿kali)-[~/Desktop/Unix_A2-main]
└─$ ./2o 2.c  1.txt
link: File exists

┌──(kali㉿kali)-[~/Desktop/Unix_A2-main]
└─$ ./2o 2.c  2.txt
Hard link created.

┌──(kali㉿kali)-[~/Desktop/Unix_A2-main]
└─$ ./2o 1.c  2.txt
link: File exists
```

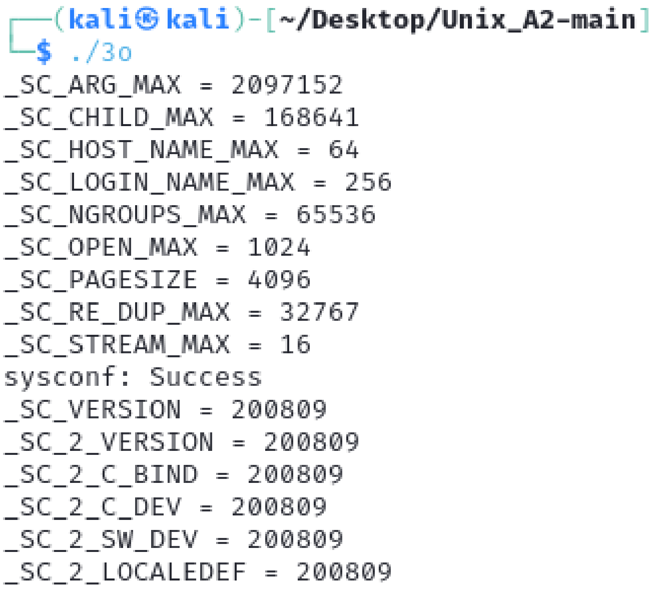
3.) Write a C/C++ POSIX compliant program that prints the POSIX defined Configuration options supported on any given system using feature test macros.

Code:



```c
#include <stdio.h>
#include <unistd.h>
#include <errno.h>

struct posix_option {
    const char *name;
    int option;
} options[] = {
    {"_SC_ARG_MAX", _SC_ARG_MAX},
    {"_SC_CHILD_MAX", _SC_CHILD_MAX},
    {"_SC_HOST_NAME_MAX", _SC_HOST_NAME_MAX},
    {"_SC_LOGIN_NAME_MAX", _SC_LOGIN_NAME_MAX},
    {"_SC_NGROUPS_MAX", _SC_NGROUPS_MAX},
    {"_SC_OPEN_MAX", _SC_OPEN_MAX},
    {"_SC_PAGESIZE", _SC_PAGESIZE},
    {"_SC_RE_DUP_MAX", _SC_RE_DUP_MAX},
    {"_SC_STREAM_MAX", _SC_STREAM_MAX},
    {"_SC_TZNAME_MAX", _SC_TZNAME_MAX},
    {"_SC_VERSION", _SC_VERSION},
    {"_SC_2_VERSION", _SC_2_VERSION},
    {"_SC_2_C_BIND", _SC_2_C_BIND},
    {"_SC_2_C_DEV", _SC_2_C_DEV},
    {"_SC_2_SW_DEV", _SC_2_SW_DEV},
    {"_SC_2_LOCALEDEF", _SC_2_LOCALEDEF}
};

int main() {
    for (size_t i = 0; i < sizeof(options) / sizeof(options[0]); ++i) {
        errno = 0;  // Reset errno before each call
        long value = sysconf(options[i].option);
        if (value == -1) {
            if (errno == EINVAL) {
                printf("%s is not supported.\n", options[i].name);
            } else {
                perror("sysconf");
            }
        } else {
            printf("%s = %ld\n", options[i].name, value);
        }
    }
    return 0;
}
```

Output:

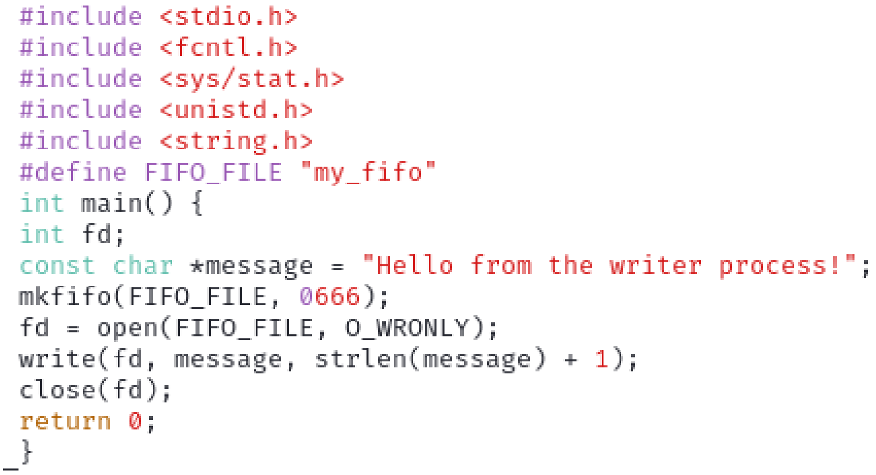


```
┌──(kali㉿kali)-[~/Desktop/Unix_A2-main]
└─$ ./3o
_SC_ARG_MAX = 2097152
_SC_CHILD_MAX = 168641
_SC_HOST_NAME_MAX = 64
_SC_LOGIN_NAME_MAX = 256
_SC_NGROUPS_MAX = 65536
_SC_OPEN_MAX = 1024
_SC_PAGESIZE = 4096
_SC_RE_DUP_MAX = 32767
_SC_STREAM_MAX = 16
sysconf: Success
_SC_VERSION = 200809
_SC_2_VERSION = 200809
_SC_2_C_BIND = 200809
_SC_2_C_DEV = 200809
_SC_2_SW_DEV = 200809
_SC_2_LOCALEDEF = 200809
```

4.) Write a C/C++ program which demonstrates Interprocess Communication between a reader process and a writer process. Use `mkfifo`, `open`, `read`, `write`, and `close` APIs in your program.

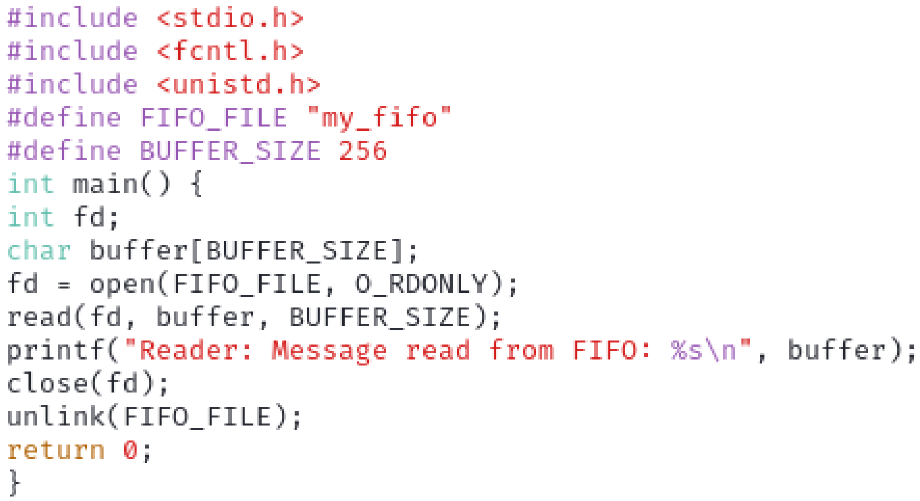
Code:

Writers part:



```
#include <stdio.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <unistd.h>
#include <string.h>
#define FIFO_FILE "my_fifo"
int main() {
int fd;
const char *message = "Hello from the writer process!";
mkfifo(FIFO_FILE, 0666);
fd = open(FIFO_FILE, O_WRONLY);
write(fd, message, strlen(message) + 1);
close(fd);
return 0;
}
```

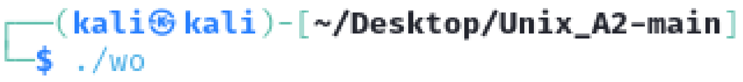
Readers Part:



```
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
#define FIFO_FILE "my_fifo"
#define BUFFER_SIZE 256
int main() {
int fd;
char buffer[BUFFER_SIZE];
fd = open(FIFO_FILE, O_RDONLY);
read(fd, buffer, BUFFER_SIZE);
printf("Reader: Message read from FIFO: %s\n", buffer);
close(fd);
unlink(FIFO_FILE);
return 0;
}
```

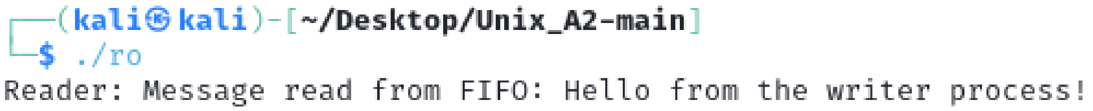
Output:

Terminal (writers)



```
┌──(kali㉿kali)-[~/Desktop/Unix_A2-main]
└─$ ./wo
```

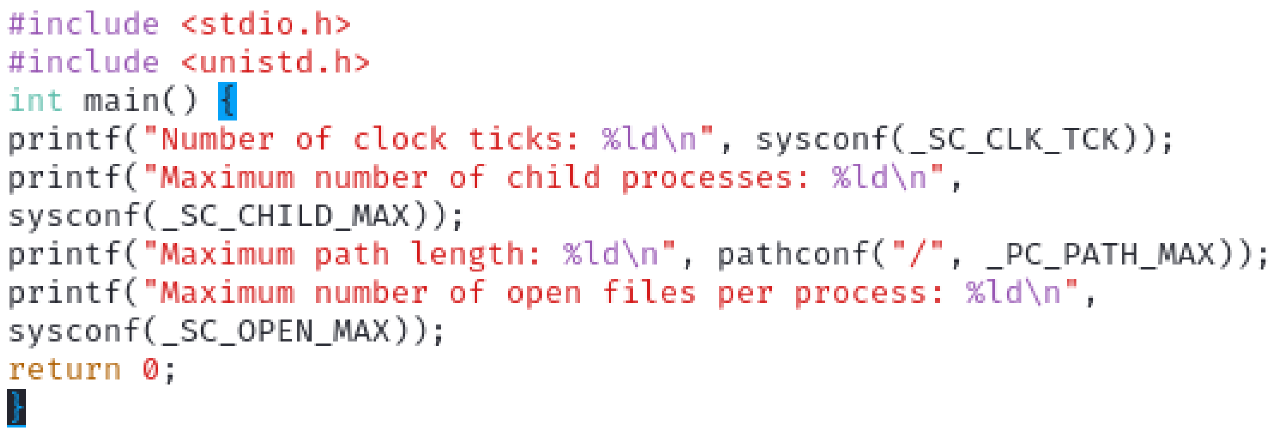
Terminal (readers)



```
┌──(kali㉿kali)-[~/Desktop/Unix_A2-main]
└─$ ./ro
Reader: Message read from FIFO: Hello from the writer process!
```

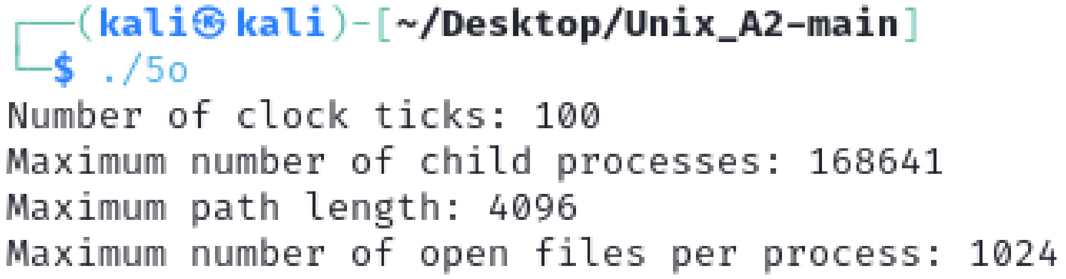
5.) Write a c or c++ program posix complement program to check following limits: i)number of clock ticks ii)Maximum number of child processes iii)Maximum path length iv) Maximum number of open files per process.

Code:



```c
#include <stdio.h>
#include <unistd.h>
int main() {
printf("Number of clock ticks: %ld\n", sysconf(_SC_CLK_TCK));
printf("Maximum number of child processes: %ld\n",
sysconf(_SC_CHILD_MAX));
printf("Maximum path length: %ld\n", pathconf("/", _PC_PATH_MAX));
printf("Maximum number of open files per process: %ld\n",
sysconf(_SC_OPEN_MAX));
return 0;
}
```

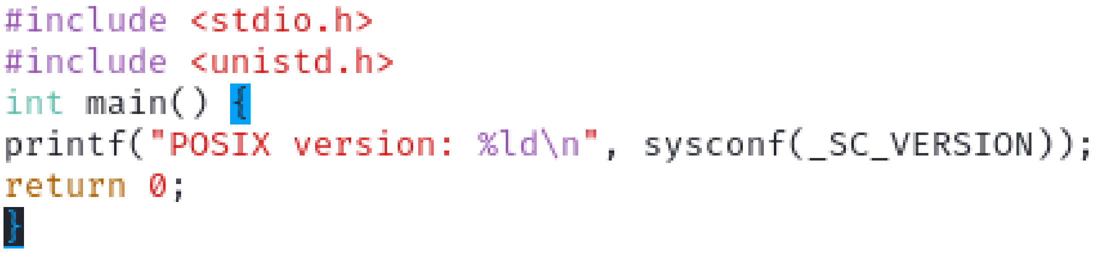
Output:



```
┌──(kali㉿kali)-[~/Desktop/Unix_A2-main]
└─$ ./5o
Number of clock ticks: 100
Maximum number of child processes: 168641
Maximum path length: 4096
Maximum number of open files per process: 1024
```

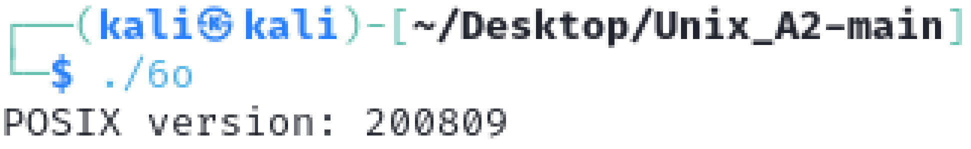
6.) Write C/C++ program to display POSIX version.

Code:



```
#include <stdio.h>
#include <unistd.h>
int main() {
printf("POSIX version: %ld\n", sysconf(_SC_VERSION));
return 0;
}
```

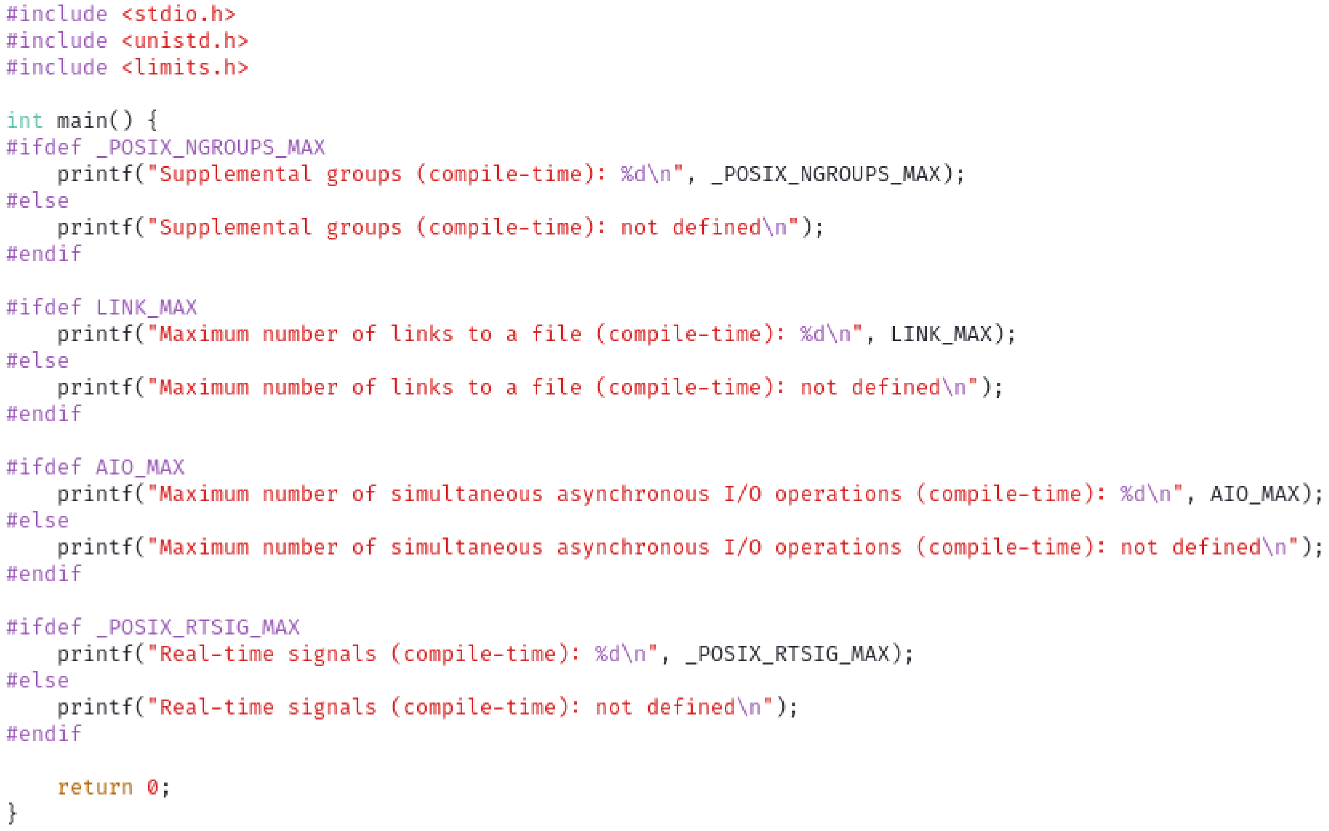
Output:



```
┌──(kali㉿kali)-[~/Desktop/Unix_A2-main]
└─$ ./6o
POSIX version: 200809
```

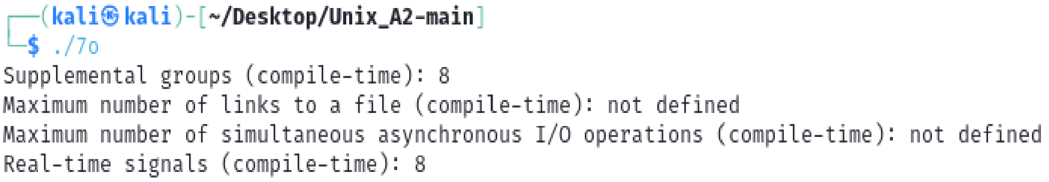
7.) Write C or C++ program to check the following compile time along with its minimum value. a)supplemental groups b)maximum number of links of a file. c)maximum number of simulate nous asynchronous I/O. d)real signals

Code:



```
#include <stdio.h>
#include <unistd.h>
#include <limits.h>

int main() {
#ifdef _POSIX_NGROUPS_MAX
    printf("Supplemental groups (compile-time): %d\n", _POSIX_NGROUPS_MAX);
#else
    printf("Supplemental groups (compile-time): not defined\n");
#endif

#ifdef LINK_MAX
    printf("Maximum number of links to a file (compile-time): %d\n", LINK_MAX);
#else
    printf("Maximum number of links to a file (compile-time): not defined\n");
#endif

#ifdef AIO_MAX
    printf("Maximum number of simultaneous asynchronous I/O operations (compile-time): %d\n", AIO_MAX);
#else
    printf("Maximum number of simultaneous asynchronous I/O operations (compile-time): not defined\n");
#endif

#ifdef _POSIX_RTSIG_MAX
    printf("Real-time signals (compile-time): %d\n", _POSIX_RTSIG_MAX);
#else
    printf("Real-time signals (compile-time): not defined\n");
#endif

    return 0;
}
```

Output:



```
┌──(kali㉿kali)-[~/Desktop/Unix_A2-main]
└─$ ./7o
Supplemental groups (compile-time): 8
Maximum number of links to a file (compile-time): not defined
Maximum number of simultaneous asynchronous I/O operations (compile-time): not defined
Real-time signals (compile-time): 8
```

8.) List the commands needed to change the following attributes. i) file size ii) user ID iii) Last access & modification time iv) hard link count

(i) Change File Size

Increase or decrease file size using the `truncate` command.

Command:

```
truncate -s <size> <filename>
```

Example:

```
truncate -s 100 filename.txt
```

(ii) Change User ID

Change the owner of a file using the `chown` command.

Command:

```
chown <new_owner> <filename>
```

Example:

```
chown newuser filename.txt
```

(iii) Change Last Access & Modification Time

Change access and modification time using the `touch` command.

Command:

```
touch -a -m -t
<[[CC]YY]MMDDhhmm[.ss]> <filename>
```

Example:

```
touch -a -m -t 202201011234
filename.txt
```

Alternatively, set time using a reference file.

Command:

```
touch -r <reference_file> <filename>
```

Example:

```
touch -r reference.txt filename.txt
```

(iv) Change Hard Link Count

Create a new hard link using the `ln` command.

Command:

```
ln <existing_file> <new_hard_link>
```

Example:

```
ln filename.txt hardlink.txt
```

Remove a hard link using the `rm` command.

Command:

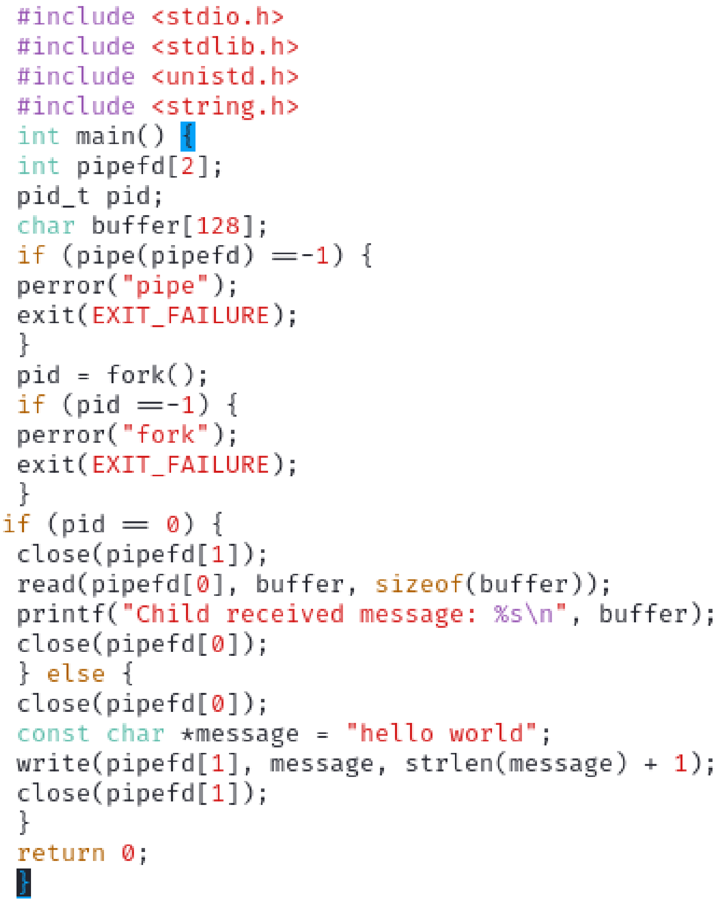
```
rm <hard_link>
```

Example:

```
m hardlink.txt
```

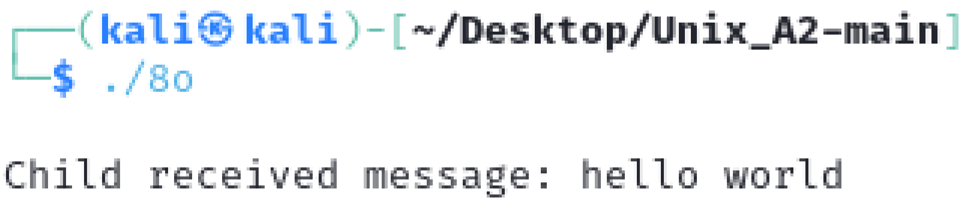
9.) Write a c program that sends "hello world" message to the child process through the pipe. The child on receiving this message should display it on the standard output.

Code:



```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
int main() {
int pipefd[2];
pid_t pid;
char buffer[128];
if (pipe(pipefd) ==-1) {
perror("pipe");
exit(EXIT_FAILURE);
}
pid = fork();
if (pid ==-1) {
perror("fork");
exit(EXIT_FAILURE);
}
if (pid == 0) {
close(pipefd[1]);
read(pipefd[0], buffer, sizeof(buffer));
printf("Child received message: %s\n", buffer);
close(pipefd[0]);
} else {
close(pipefd[0]);
const char *message = "hello world";
write(pipefd[1], message, strlen(message) + 1);
close(pipefd[1]);
}
return 0;
}
```

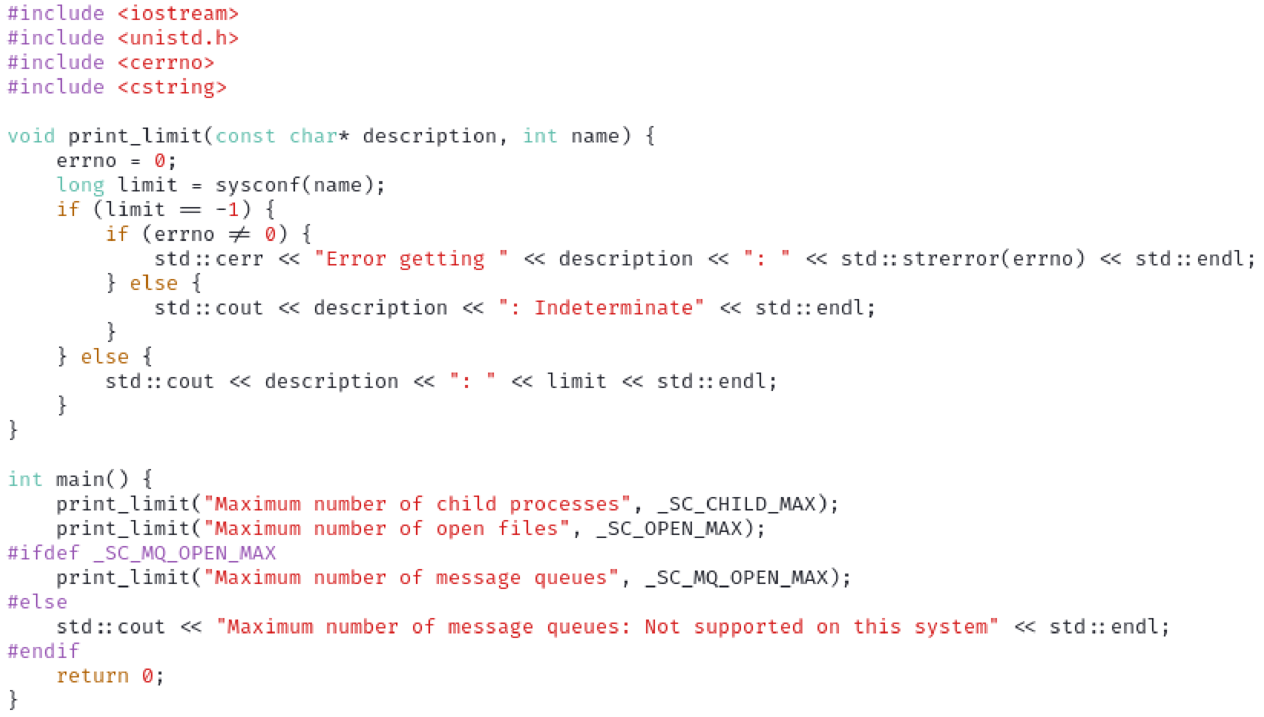
Output:



```
┌──(kali㉿kali)-[~/Desktop/Unix_A2-main]
└─$ ./8o

Child received message: hello world
```

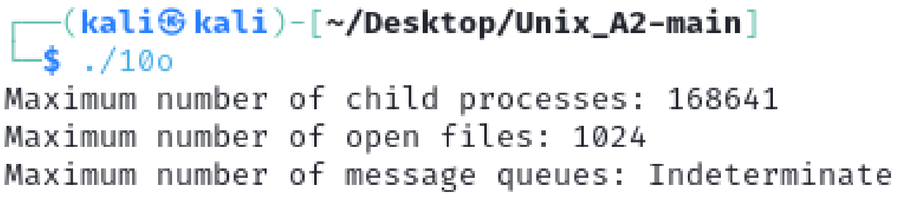
10.) Write a c++ program to list the actual values of the following system configuration limits on a given UNIX OS. i) Maximum no. of child processes that can be created. ii) Maximum no. of files that can be opened simultaneously. iii) Maximum no. of message queues that can be accessed

Code:



```
#include <iostream>
#include <unistd.h>
#include <cerrno>
#include <cstring>

void print_limit(const char* description, int name) {
    errno = 0;
    long limit = sysconf(name);
    if (limit == -1) {
        if (errno != 0) {
            std::cerr << "Error getting " << description << ": " << std::strerror(errno) << std::endl;
        } else {
            std::cout << description << ": Indeterminate" << std::endl;
        }
    } else {
        std::cout << description << ": " << limit << std::endl;
    }
}

int main() {
    print_limit("Maximum number of child processes", _SC_CHILD_MAX);
    print_limit("Maximum number of open files", _SC_OPEN_MAX);
#ifdef _SC_MQ_OPEN_MAX
    print_limit("Maximum number of message queues", _SC_MQ_OPEN_MAX);
#else
    std::cout << "Maximum number of message queues: Not supported on this system" << std::endl;
#endif
    return 0;
}
```

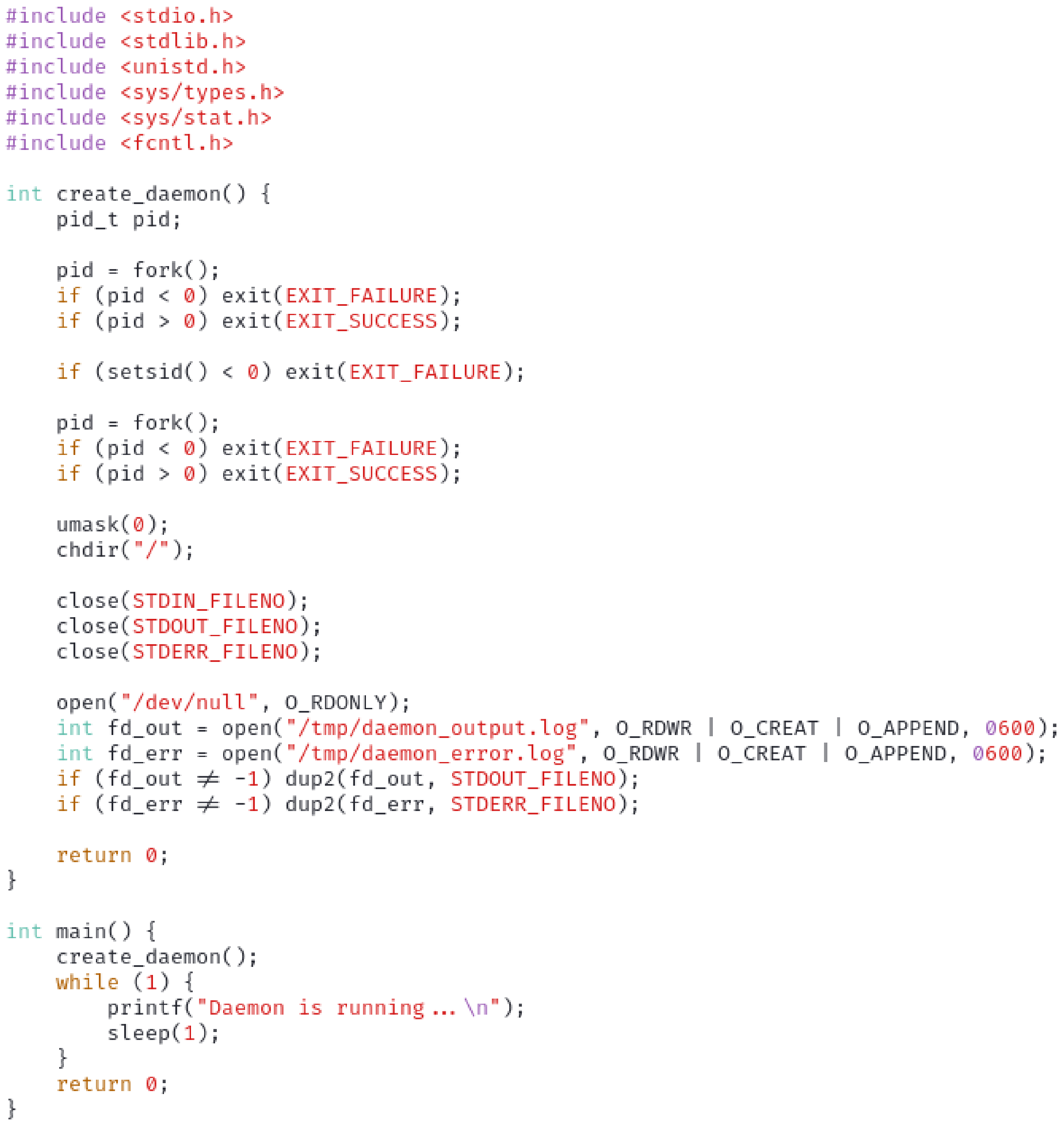
Output:



```
┌──(kali㉿kali)-[~/Desktop/Unix_A2-main]
└─$ ./10o
Maximum number of child processes: 168641
Maximum number of open files: 1024
Maximum number of message queues: Indeterminate
```

11.) Write a program to transform a normal user process into a daemon process.

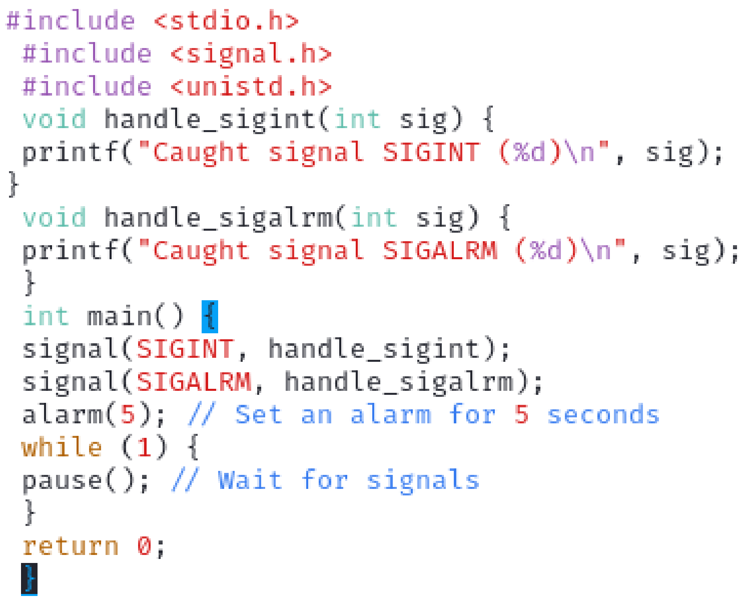
Code:



```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

int create_daemon() {
    pid_t pid;

    pid = fork();
    if (pid < 0) exit(EXIT_FAILURE);
    if (pid > 0) exit(EXIT_SUCCESS);

    if (setsid() < 0) exit(EXIT_FAILURE);

    pid = fork();
    if (pid < 0) exit(EXIT_FAILURE);
    if (pid > 0) exit(EXIT_SUCCESS);

    umask(0);
    chdir("/");

    close(STDIN_FILENO);
    close(STDOUT_FILENO);
    close(STDERR_FILENO);

    open("/dev/null", O_RDONLY);
    int fd_out = open("/tmp/daemon_output.log", O_RDWR | O_CREAT | O_APPEND, 0600);
    int fd_err = open("/tmp/daemon_error.log", O_RDWR | O_CREAT | O_APPEND, 0600);
    if (fd_out != -1) dup2(fd_out, STDOUT_FILENO);
    if (fd_err != -1) dup2(fd_err, STDERR_FILENO);

    return 0;
}

int main() {
    create_daemon();
    while (1) {
        printf("Daemon is running...\n");
        sleep(1);
    }
    return 0;
}
```

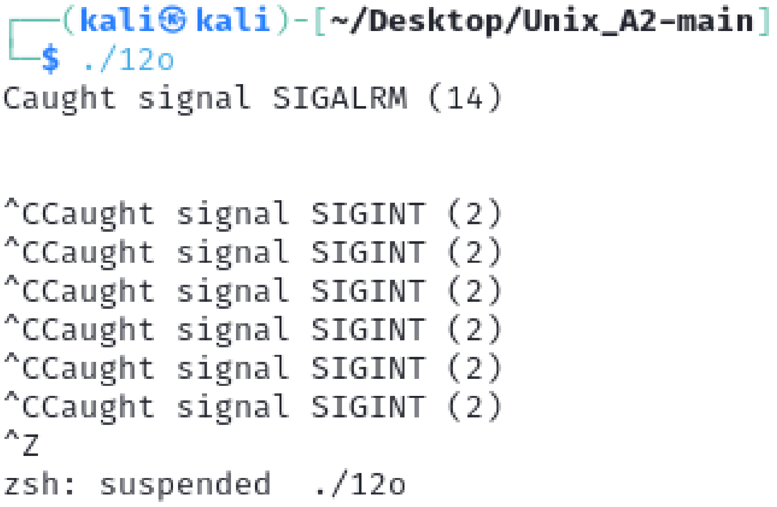
12.) Write a program to setup signals handlers for SIGINT & SIGACRAM signals.

Code:



```
#include <stdio.h>
 #include <signal.h>
 #include <unistd.h>
 void handle_sigint(int sig) {
 printf("Caught signal SIGINT (%d)\n", sig);
}
 void handle_sigalrm(int sig) {
 printf("Caught signal SIGALRM (%d)\n", sig);
 }
 int main() {
 signal(SIGINT, handle_sigint);
 signal(SIGALRM, handle_sigalrm);
 alarm(5); // Set an alarm for 5 seconds
 while (1) {
 pause(); // Wait for signals
 }
 return 0;
 }
```

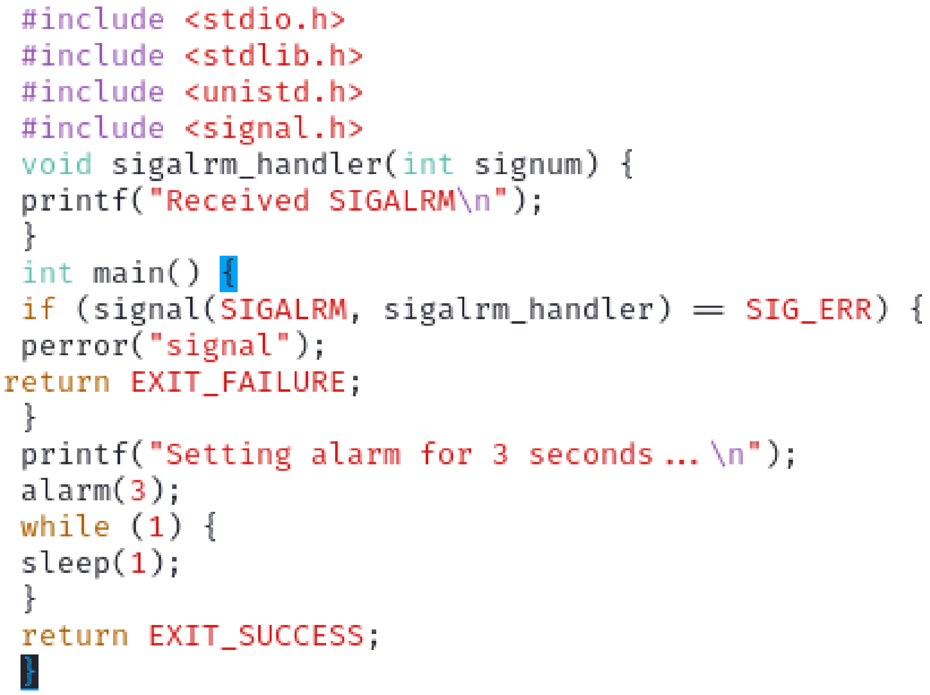
Output:



```
┌──(kali㉿kali)-[~/Desktop/Unix_A2-main]
└─$ ./12o
Caught signal SIGALRM (14)


^CCaught signal SIGINT (2)
^CCaught signal SIGINT (2)
^CCaught signal SIGINT (2)
^CCaught signal SIGINT (2)
^CCaught signal SIGINT (2)
^CCaught signal SIGINT (2)
^Z
zsh: suspended  ./12o
```

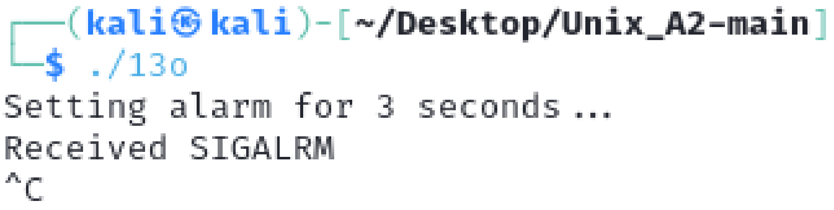
13.) Write a C/C++ program to show the use of alarm.

Code:



```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <signal.h>
void sigalrm_handler(int signum) {
printf("Received SIGALRM\n");
}
int main() {
if (signal(SIGALRM, sigalrm_handler) == SIG_ERR) {
perror("signal");
return EXIT_FAILURE;
}
printf("Setting alarm for 3 seconds...\n");
alarm(3);
while (1) {
sleep(1);
}
return EXIT_SUCCESS;
}
```

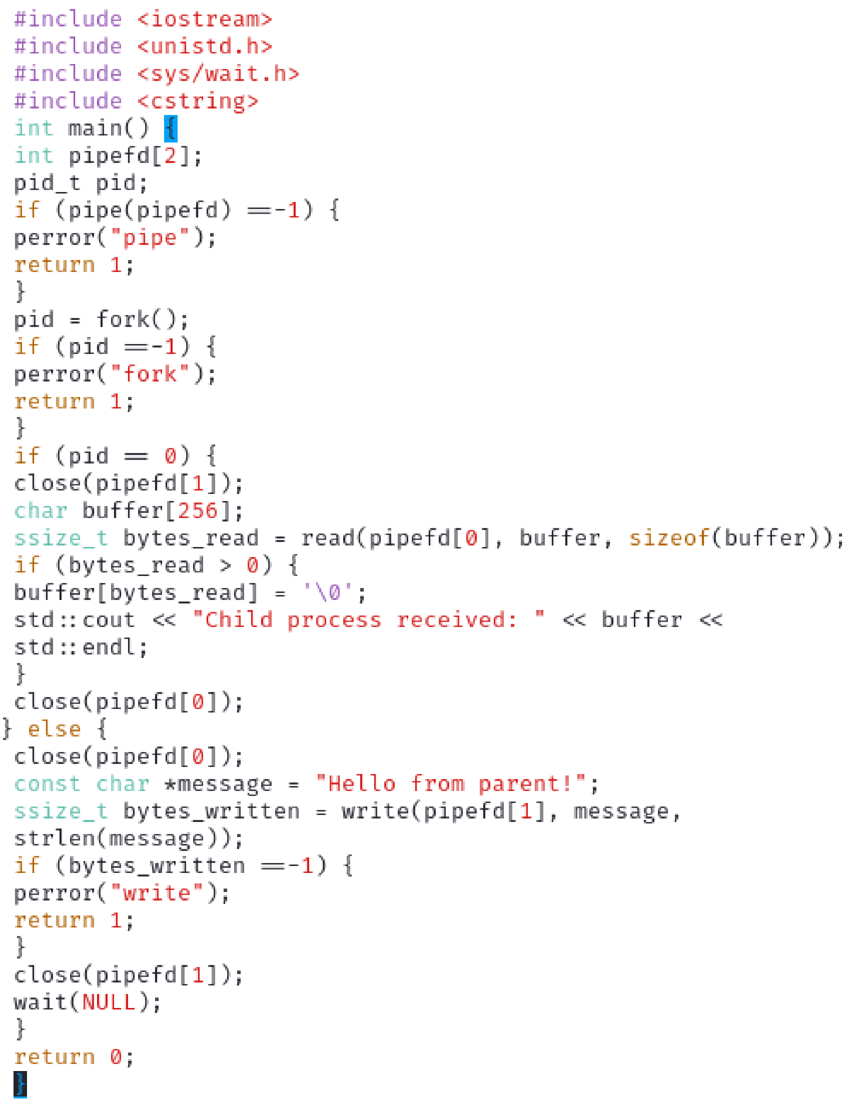
Output:



```
┌──(kali㉿kali)-[~/Desktop/Unix_A2-main]
└─$ ./13o
Setting alarm for 3 seconds...
Received SIGALRM
^C
```

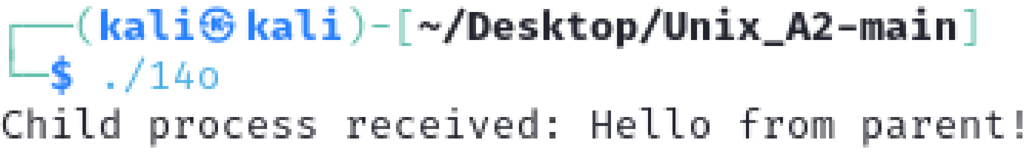
14.) Write a C++ program to send data from parent to child over a pipe.

Code:



```
#include <iostream>
#include <unistd.h>
#include <sys/wait.h>
#include <cstring>
int main() {
int pipefd[2];
pid_t pid;
if (pipe(pipefd) ==-1) {
perror("pipe");
return 1;
}
pid = fork();
if (pid ==-1) {
perror("fork");
return 1;
}
if (pid == 0) {
close(pipefd[1]);
char buffer[256];
ssize_t bytes_read = read(pipefd[0], buffer, sizeof(buffer));
if (bytes_read > 0) {
buffer[bytes_read] = '\0';
std::cout << "Child process received: " << buffer <<
std::endl;
}
close(pipefd[0]);
} else {
close(pipefd[0]);
const char *message = "Hello from parent!";
ssize_t bytes_written = write(pipefd[1], message,
strlen(message));
if (bytes_written ==-1) {
perror("write");
return 1;
}
close(pipefd[1]);
wait(NULL);
}
return 0;
}
```

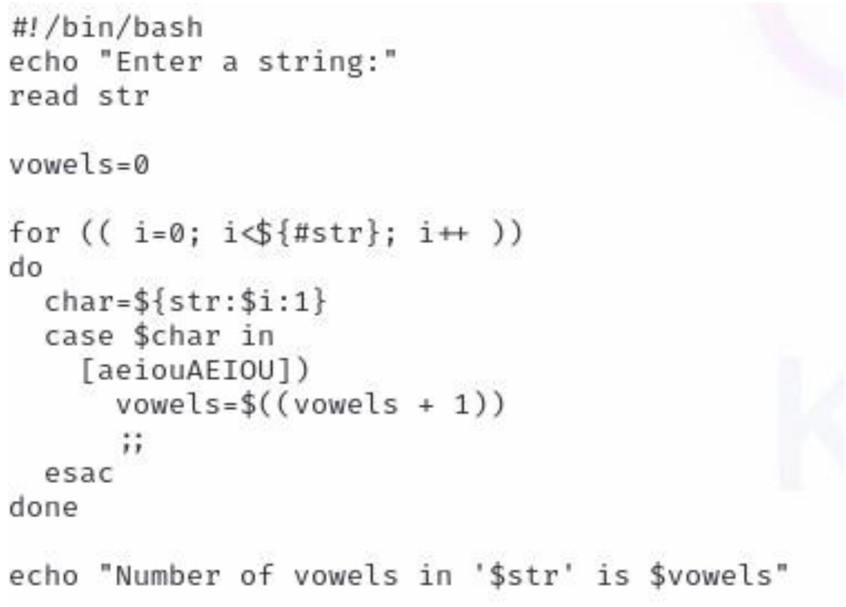
Output:



```
┌──(kali㉿kali)-[~/Desktop/Unix_A2-main]
└─$ ./14o
Child process received: Hello from parent!
```

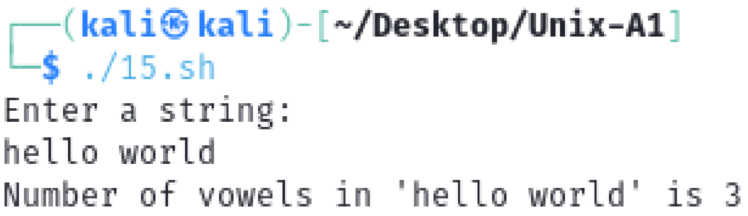
15.) Shell script to count the number of vowels of a string.

Code:



```
#!/bin/bash
echo "Enter a string:"
read str

vowels=0

for (( i=0; i<${#str}; i++ ))
do
  char=${str:$i:1}
  case $char in
    [aeiouAEIOU])
      vowels=$((vowels + 1))
      ;;
  esac
done

echo "Number of vowels in '$str' is $vowels"
```

Output:



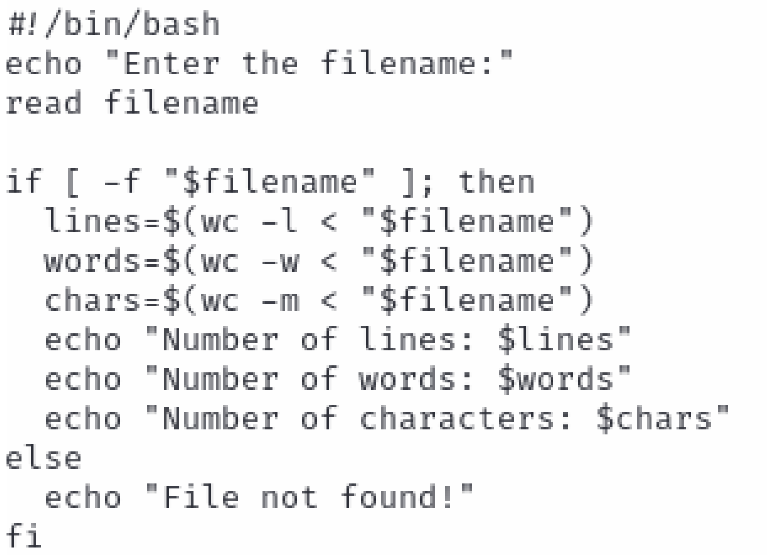
```
┌──(kali㉿kali)-[~/Desktop/Unix-A1]
└─$ ./15.sh
Enter a string:
hello world
Number of vowels in 'hello world' is 3
```

Description:

This script counts the number of vowels in a given string. The user inputs a string, and the script iterates through each character, counting the vowels (a, e, i, o, u). The total number of vowels is then printed.

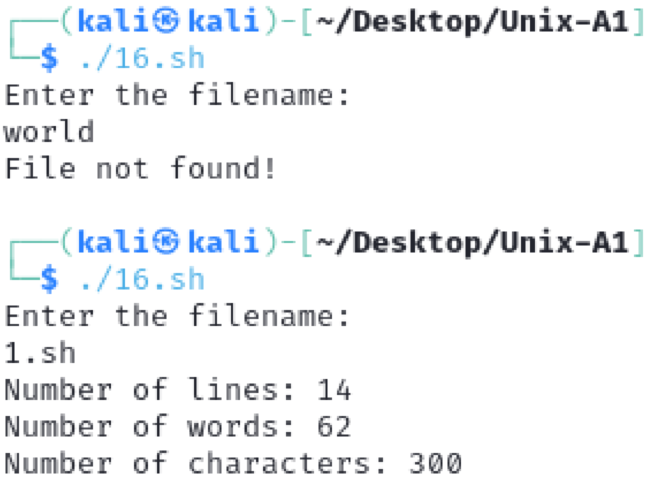
16.) Shell script to check number of lines, words, characters in a file.

Code:



```
#!/bin/bash
echo "Enter the filename:"
read filename

if [ -f "$filename" ]; then
  lines=$(wc -l < "$filename")
  words=$(wc -w < "$filename")
  chars=$(wc -m < "$filename")
  echo "Number of lines: $lines"
  echo "Number of words: $words"
  echo "Number of characters: $chars"
else
  echo "File not found!"
fi
```

Output:



```
┌──(kali㉿kali)-[~/Desktop/Unix-A1]
└─$ ./16.sh
Enter the filename:
world
File not found!

┌──(kali㉿kali)-[~/Desktop/Unix-A1]
└─$ ./16.sh
Enter the filename:
1.sh
Number of lines: 14
Number of words: 62
Number of characters: 300
```

Description:

The script reads a file and counts the number of lines, words, and characters within it. The user provides the file name, and the script uses commands to calculate these counts. The results, showing the number of lines, words, and characters, are then printed.