

Advanced Unix Programming Lab 4

Purva Tendulkar : 111403049

Shruti Dogra : 111403075

Q1. Print all existing environment variables with their values. Later input a new variable and its value and add to the environment list. Once again print the list.

Code :

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 20

extern char **environ;

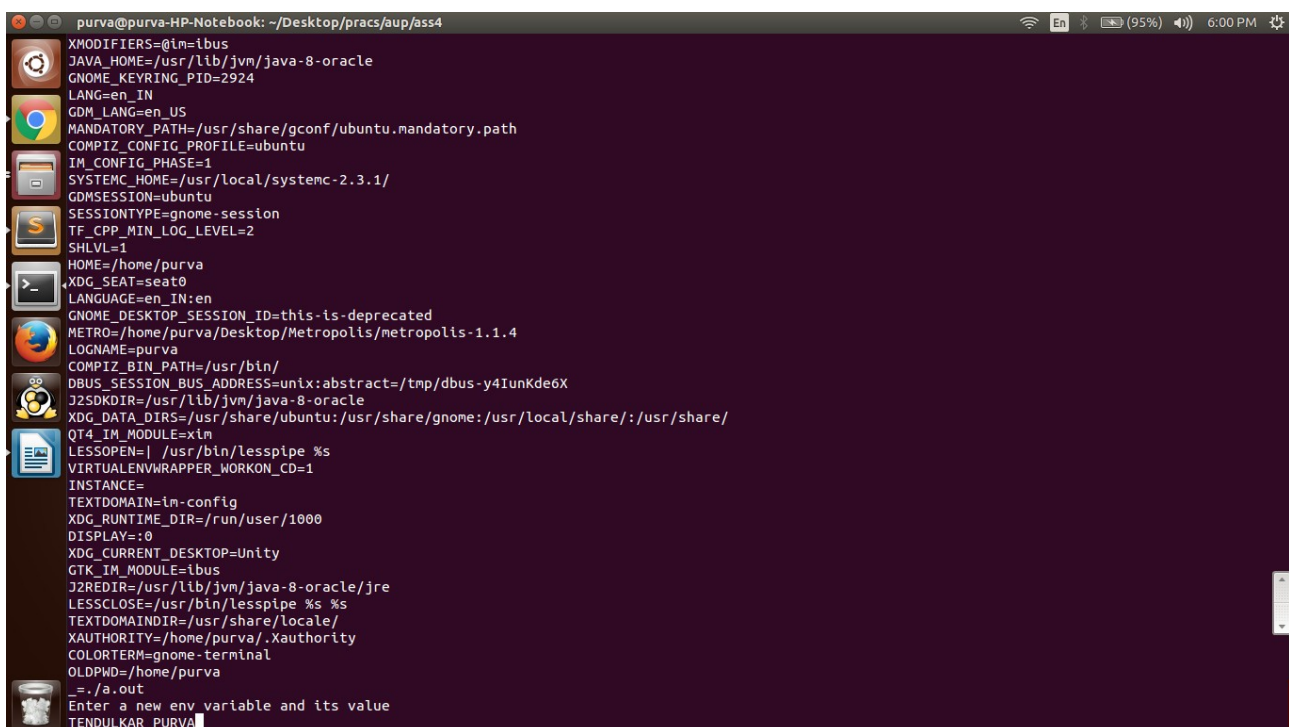
int main() {
    char name[MAX], val[MAX], **env;

    for (env = environ; *env; ++env)
        printf("%s\n", *env);

    printf("Enter a new env variable and its value\n");
    scanf("%s%s", name, val);
    setenv(name, val, 1);

    for (env = environ; *env; ++env)
        printf("%s\n", *env);
    return 0;
}
```

Input and Output Screenshots :

A screenshot of a terminal window on a Linux system. The window title is 'purva@purva-HP-Notebook: ~/Desktop/pracs/aup/ass4'. The terminal displays a list of environment variables, including XMODIFIERS, JAVA_HOME, GNOME_KEYRING_PID, LANG, GDM_LANG, MANDATORY_PATH, COMPIZ_CONFIG_PROFILE, IM_CONFIG_PHASE, SYSTEMC_HOME, GDMSESSION, SESSIONTYPE, TF_CPP_MIN_LOG_LEVEL, SHLVL, HOME, XDG_SEAT, LANGUAGE, GNOME_DESKTOP_SESSION_ID, METRO, LOGNAME, COMPIZ_BIN_PATH, DBUS_SESSION_BUS_ADDRESS, J2SDKDIR, XDG_DATA_DIRS, QT4_IM_MODULE, LESSOPEN, VIRTUALENVWRAPPER_WORKON_CD, INSTANCE, TEXTDOMAIN, XDG_RUNTIME_DIR, DISPLAY, XDG_CURRENT_DESKTOP, GTK_IM_MODULE, J2REDIR, LESSCLOSE, TEXTDOMAINDIR, XAUTHORITY, COLORTERM, OLDPWD, and _=. The prompt shows the user has entered 'TENDULKAR PURVA' and the terminal is waiting for a value. The terminal output shows the list of environment variables, followed by the prompt 'Enter a new env variable and its value', and then the user input 'TENDULKAR PURVA'. The terminal window has a dark background and a light-colored text. The status bar at the bottom shows the battery level at 95% and the time as 6:00 PM.

```
purva@purva-HP-Notebook: ~/Desktop/pracs/aup/ass4
XMODIFIERS=@im=ibus
JAVA_HOME=/usr/lib/jvm/java-8-oracle
GNOME_KEYRING_PID=2924
LANG=en_IN
GDM_LANG=en_US
MANDATORY_PATH=/usr/share/gconf/ubuntu.mandatory.path
COMPIZ_CONFIG_PROFILE=ubuntu
IM_CONFIG_PHASE=1
SYSTEMC_HOME=/usr/local/systemc-2.3.1/
GDMSESSION=ubuntu
SESSIONTYPE=gnome-session
TF_CPP_MIN_LOG_LEVEL=2
SHLVL=1
HOME=/home/purva
XDG_SEAT=seat0
LANGUAGE=en_IN:en
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
METRO=/home/purva/Desktop/Metropolis/metropolis-1.1.4
LOGNAME=purva
COMPIZ_BIN_PATH=/usr/bin/
DBUS_SESSION_BUS_ADDRESS=unix:abstract=/tmp/dbus-y4IunKde6X
J2SDKDIR=/usr/lib/jvm/java-8-oracle
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/share/gnome:/usr/local/share:/usr/share/
QT4_IM_MODULE=xim
LESSOPEN=| /usr/bin/lesspipe %s
VIRTUALENVWRAPPER_WORKON_CD=1
INSTANCE=
TEXTDOMAIN=tm-config
XDG_RUNTIME_DIR=/run/user/1000
DISPLAY=:0
XDG_CURRENT_DESKTOP=Unity
GTK_IM_MODULE=ibus
J2REDIR=/usr/lib/jvm/java-8-oracle/jre
LESSCLOSE=/usr/bin/lesspipe %s %s
TEXTDOMAINDIR=/usr/share/locale/
XAUTHORITY=/home/purva/.Xauthority
COLORTERM=gnome-terminal
OLDPWD=/home/purva
_=/a.out
TENDULKAR=PURVA
purva@purva-HP-Notebook: ~/Desktop/pracs/aup/ass4$
```

Explanation :

The new environment variable here is “TENDULKAR” and the corresponding value is “PURVA”.

Q2. With appropriate comments write a program using setjmp and longjmp to verify the status of different types of variables after invoking longjmp.

Code :

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

```
static int globval; //Global Variable
jmp_buf jmpbuffer;
```

```
void f2() {
    longjmp(jmpbuffer, 1);
```

```
/* While invoking this longjmp, the variables that are stored in memory will have values
as of the time of longjmp, while variables
stored in registers are restored to their
values when setjmp was called.
Therefore values of autoval and regival get changed to 2 and 3 respectively whereas
the rest remain the same */
```

```
}
```

```
void f1(int i, int j, int k, int l) {
    printf("\nIn f1(): ");
    printf("globval = %d, autoval = %d, regival = %d, volaval = %d, statval = %d\n\n", globval,
        i, j, k, l);
```

```
/* In the above printf, the values of variables will be the same as passed to the function
f1 as longjmp hasnt been invoked yet */
```

```

    f2();
}

int main() {
    int autoval;                //Automatic Variable
    register int regival;       //Register Variable
    int volatile volaval;       //Volatile Variable
    static int statval;         //Static Variable
    globval = 1; autoval = 2; regival = 3; volaval = 4; statval = 5;

    /* The value of all types of variables are set before setjmp. When compiled using
    optimization (-O), AUTOMATIC and REGISTER variables get stored in the register. Rest
    get stored in memory. When compiled without optimization, all go to memory. */

    printf("\nBefore setjmp: ");
    printf("globval = %d, autoval = %d, regival = %d, volaval = %d, statval = %d\n\n",
           globval, autoval, regival, volaval, statval);

    if (setjmp(jmpbuffer) != 0) {
        printf("\nAfter longjmp: ");
        printf("globval = %d, autoval = %d, regival = %d, volaval = %d, statval = %d\n\n",
               globval, autoval, regival, volaval, statval);
        exit(0);
    }

    globval = 95, autoval = 96, regival = 97, volaval = 98, statval = 99;
    f1(autoval, regival, volaval, statval);
    exit(0);
}

```

Input and Output Screenshots :

```

purva@purva-HP-Notebook: ~/Desktop/pracs/aup/ass4
purva@purva-HP-Notebook:~/Desktop/pracs/aup/ass4$ gcc test2.c -Wall
purva@purva-HP-Notebook:~/Desktop/pracs/aup/ass4$ ./a.out
Before setjmp: globval = 1, autoval = 2, regival = 3, volaval = 4, statval = 5
In f1(): globval = 95, autoval = 96, regival = 97, volaval = 98, statval = 99
After longjmp: globval = 95, autoval = 96, regival = 97, volaval = 98, statval = 99
purva@purva-HP-Notebook:~/Desktop/pracs/aup/ass4$ gcc -O test2.c -Wall
purva@purva-HP-Notebook:~/Desktop/pracs/aup/ass4$ ./a.out
Before setjmp: globval = 1, autoval = 2, regival = 3, volaval = 4, statval = 5
In f1(): globval = 95, autoval = 96, regival = 97, volaval = 98, statval = 99
After longjmp: globval = 95, autoval = 2, regival = 3, volaval = 98, statval = 99
purva@purva-HP-Notebook:~/Desktop/pracs/aup/ass4$

```

Q3. Measures the performance of the getpid() and the fork functions using gettimeofday to measure the the execution time. Measure the performance ten times for each of the two system calls in the program itself and provide the timing results and compute an average for each system call.

Code :

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
#include <time.h>
#include <unistd.h>
#define N 10

double get_time_difference(struct timeval start, struct timeval end) {
    return (end.tv_sec - start.tv_sec) + (end.tv_usec - start.tv_usec) / 1000000.0;
}

double time_getpid() {
    printf("getpid() function calls...\n");
    struct timeval getpid_start;
    struct timeval getpid_end;
    double sum_getpid_times_elapsed = 0.0;
    pid_t pid;
    int i;

    for(i = 0; i < N; i++){
        gettimeofday(&getpid_start, NULL);
        pid = getpid();
        gettimeofday(&getpid_end, NULL);
        sum_getpid_times_elapsed += get_time_difference(getpid_start, getpid_end);
        printf("%d - %lf\n", i, get_time_difference(getpid_start, getpid_end));
    }
    return sum_getpid_times_elapsed / 10;
}

double time_fork() {
    printf("fork() function calls...\n");
    struct timeval fork_start;
    struct timeval fork_end;
    double sum_fork_times_elapsed = 0.0;
    int i;

    for (i = 0; i < N; i++) {
        gettimeofday(&fork_start, NULL);
        int p = fork();
        gettimeofday(&fork_end, NULL);
        if (p) {
            sum_fork_times_elapsed += get_time_difference(fork_start, fork_end);
            printf("%d - %lf\n", i, get_time_difference(fork_start, fork_end));
        }
        else
            exit(0);
    }
}
```

```

    }
    return sum_fork_times_elapsed / 10;
}

int main(){
    printf("-----\n");
    printf("AVERAGE TIME FOR getpid() - %lf\n", time_getpid());
    printf("-----\n");
    printf("AVERAGE TIME FOR fork() - %lf\n", time_fork());
    printf("-----\n");
    return 0;
}

```

Input and Output Screenshots :

The screenshot shows a terminal window on a Linux system. The user has compiled a C program named 'test3.c' using 'gcc' and executed it with './a.out'. The program outputs the average time for 'getpid()' and 'fork()' system calls over 10 iterations. The output is as follows:

```

purva@purva-HP-Notebook: ~/Desktop/pracs/aup/ass4
purva@purva-HP-Notebook:~/Desktop/pracs/aup/ass4$ gcc test3.c
purva@purva-HP-Notebook:~/Desktop/pracs/aup/ass4$ ./a.out
-----
getpid() function calls...
0 - 0.000004
1 - 0.000000
2 - 0.000000
3 - 0.000000
4 - 0.000000
5 - 0.000001
6 - 0.000001
7 - 0.000001
8 - 0.000000
9 - 0.000000
AVERAGE TIME FOR getpid() - 0.000001
-----
fork() function calls...
0 - 0.000091
1 - 0.000123
2 - 0.000110
3 - 0.000098
4 - 0.000104
5 - 0.000089
6 - 0.000096
7 - 0.000086
8 - 0.000096
9 - 0.000078
AVERAGE TIME FOR fork() - 0.000097
-----
purva@purva-HP-Notebook:~/Desktop/pracs/aup/ass4$

```