

AUP : Assignment - 4 [Process Control]

Aditi Rajendra Medhane 111803177

20th September 2021

Q2

Write a program to print all existing environment variables with their values. Later input a new variable and its value and add to the environment list. Also change the value of PATH to “/usr/bin”. Once again call to print the environment list.

Then in the command line, print the environment list. What is your observation?

Code

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <pwd.h>
4  #include <sys/types.h>
5  #include <sys/stat.h>
6  #include <unistd.h>
7  #include <fcntl.h>
8  #include <errno.h>
9  #include <dirent.h>
10
11 void print_env(char **envp){
12     while(*envp){
13         printf("%s\n", *envp);
14         envp++;
15     }
16 }
17
18 int main(){
19     extern char **environ;
20
21     print_env(environ);
22     printf("\n");
23
24     if(putenv("PATH=/usr/bin") == -1){
25         perror("putenv");
26         return errno;
27     }
28
29     printf("Changed PATH\n");
30
31     print_env(environ);
32     return 0;
33 }
34
```

Explanation

- At the start of execution, **PATH** environment variable has some value
- After a successful call to *putenv* changes the value of **PATH** to */usr/bin*
- After the program exits, the value of the path when *env* command is run is still the initial value.
- This is because any program inherits the environment variables of it's parent. But when a child changes the value of it's environment, the value of *environ* for it's parent does not change.

Output

```
Activities  Terminal  Sep 20 11:01 AM
hp@aditi: ~/Desktop/BTech/AUP/LAB/4

hp@aditi: ~/Desktop/BTech/TEST_CODE/aup-main/...  hp@aditi: ~/Desktop/BTech/AUP/LAB/4  hp@aditi: ~/Desktop/BTech/AUP/LAB/3
1;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:*.m
4a=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=0
0;36:
XDG_CURRENT_DESKTOP=ubuntu:GNOME
VTE_VERSION=6003
GNOME_TERMINAL_SCREEN=/org/gnome/Terminal/screen/ea3dd8bd_2de5_4a72_9d39_a8bdca0dfe1e
INVOCATION_ID=b98428d08f4c47e7982726d3c3c74f96
MANAGERPID=6734
GJS_DEBUG_OUTPUT=stderr
LESSCLOSE=/usr/bin/lesspipe %s %s
XDG_SESSION_CLASS=user
TERM=xterm-256color
DEFAULTS_PATH=/usr/share/gconf/ubuntu.default.path
LESSOPEN=| /usr/bin/lesspipe %s
USER=hp
GNOME_TERMINAL_SERVICE=:1.86
DISPLAY=:0
SHLVL=1
QT_IM_MODULE=ibus
XDG_RUNTIME_DIR=/run/user/1000
JOURNAL_STREAM=9:45133
XDG_DATA_DIRS=/usr/share/ubuntu:/home/hp/.local/share/flatpak/exports/share:/var/lib/flatpak/exports/share:/usr/local/share:/usr/share:/var/lib/snap
d/desktop
PATH=/home/hp/.local/bin:/home/hp/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
GDMSESSION=ubuntu
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
OLDPWD=/home/hp/Desktop/BTech
_=/2

Changed PATH
SHELL=/bin/bash
SESSION_MANAGER=local/aditi:@/tmp/.ICE-unix/6987,unix/aditi:/tmp/.ICE-unix/6987
QT_ACCESSIBILITY=1
QT_FONTCONFIG_PATH=/usr/share/fonts
QT_FONTCONFIG_PATH=/usr/share/fonts
```

Figure 1: Value of PATH variable printed in program before it is changed

```

Activities  Terminal  Sep 20 11:03 AM
hp@aditi: ~/Desktop/BTech/AUP/LAB/4

hp@aditi: ~/Desktop/BTech/TEST_CODE/aup-main/...  hp@aditi: ~/Desktop/BTech/AUP/LAB/4  hp@aditi: ~/Desktop/BTech/AUP/LAB/3

INVOCATION_ID=b98428d08f4c47e7982726d3c3c74f96
MANAGERPID=6734
GJS_DEBUG_OUTPUT=stderr
LESSCLOSE=/usr/bin/lesspipe %s %s
XDG_SESSION_CLASS=user
TERM=xterm-256color
DEFAULTS_PATH=/usr/share/gconf/ubuntu.default.path
LESSOPEN=| /usr/bin/lesspipe %s
USER=hp
GNOME_TERMINAL_SERVICE=:1.86
DISPLAY=:0
SHLVL=1
QT_IM_MODULE=ibus
XDG_RUNTIME_DIR=/run/user/1000
JOURNAL_STREAM=9:45133
XDG_DATA_DIRS=/usr/share/ubuntu:/home/hp/.local/share/flatpak/exports/share:/var/lib/flatpak/exports/share:/usr/local/share:/usr/share:/var/lib/snapd/desktop
PATH=/usr/bin
GDMSESSION=ubuntu
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
OLDPWD=/home/hp/Desktop/BTech
_=/2
hp@aditi:~/Desktop/BTech/AUP/LAB/4$ ./2 | grep PATH
MANDATORY_PATH=/usr/share/gconf/ubuntu.mandatory.path
WINDOWPATH=2
DEFAULTS_PATH=/usr/share/gconf/ubuntu.default.path
PATH=/home/hp/.local/bin:/home/hp/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
Changed PATH
MANDATORY_PATH=/usr/share/gconf/ubuntu.mandatory.path
WINDOWPATH=2
DEFAULTS_PATH=/usr/share/gconf/ubuntu.default.path
PATH=/usr/bin
hp@aditi:~/Desktop/BTech/AUP/LAB/4$

```

Figure 2: Value of PATH variable after it is changed using putenv + GREP “PATH”

Q3

Write a program to include different types of variables to demonstrate the behavior of setjmp/longjmp.

- Include a public variable
- Include a jmp_buf automatic variable, a static variable and automatic in main()
- Invoke a function a() with the argument jmp_buf variable.
- If return values of a() is nonzero, exit.
- Update values of public, static and automatic variables
- Then invoke b() with the argument jmp_buf variable.
- Print values of public, static and automatic variables
- In a(),
 - Include a static variable and automatic variable
 - setjmp() invocation with argument as received jmp_buf variable.
 - Update values of static variable and automatic variable
 - Return value of return value of setjmp
- In b(), just invoke longjmp() with received jmp_buf variable and a non zero value.

Code

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <pwd.h>
4  #include <sys/types.h>
5  #include <sys/stat.h>
6  #include <unistd.h>
7  #include <fcntl.h>
8  #include <errno.h>
9  #include <dirent.h>
10
11 #include <setjmp.h>
12
13 int public_var;
14 jmp_buf buf;
15 static int static_var;
16
17 int a(jmp_buf buf) {
18     static int a_static_var;

```

```

19     auto int a_auto_var;
20     int ret;
21
22     a_static_var = 4;
23     a_auto_var = 5;
24
25     printf("a_static_var = %d\na_auto_var = %d\npublic_var = %d\nstatic_var = %d\n\n",
26           a_static_var, a_auto_var, public_var, static_var);
27
28     ret = setjmp(buf);
29
30     printf("setjump(buf) = %d\n", ret);
31     printf("a_static_var = %d\na_auto_var = %d\npublic_var = %d\nstatic_var = %d\n\n",
32           a_static_var, a_auto_var, public_var, static_var);
33
34     a_static_var = 104;
35     a_auto_var = 105;
36
37     return ret;
38 }
39
40 int b(jmp_buf buf) {
41     longjmp(buf, 42);
42 }
43
44
45 int main(void) {
46     auto int auto_var;
47
48     public_var = 1;
49     static_var = 2;
50     auto_var = 3;
51
52     printf("Before a(buf)\n");
53     printf("public_var = %d\nstatic_var = %d\nauto_var = %d\n\n",
54           public_var, static_var, auto_var);
55
56     if (a(buf)) {
57         exit(0);
58     }
59
60     printf("After a(buf), before b(buf)\n");
61     printf("public_var = %d\nstatic_var = %d\nauto_var = %d\n\n",
62           public_var, static_var, auto_var);
63
64     public_var = 101;
65     static_var = 102;
66     auto_var = 103;
67
68     b(buf);
69
70     printf("After b(buf)\n");
71     printf("public_var = %d\nstatic_var = %d\nauto_var = %d\n\n",
72           public_var, static_var, auto_var);
73
74     return 0;
75 }

```

Explanation

In case of the program above, the values of the static and global variables will *not* be rolled back to initial state. As for the *automatic* variables..

To quote the manual page for *longjmp*-

the values of automatic variables are unspecified after a call to `longjmp()` if they meet all the following criteria:

- they are local to the function that made the corresponding `setjmp()` call;
- their values are changed between the calls to `setjmp()` and `longjmp()`; and
- they are not declared as volatile.

It is observed that for an unoptimized code, the value of the automatic variable `a__auto_var` is *not* rolled back. This is because it is not stored in a register.

For the optimized code, the automatic variable value gets rolled back- this is because `a__auto_var` is stored in the register.

Note that the output itself is different- this is because `clang` and `gcc` might eliminate sections of code which they deem useless.

Output

```

Activities  Terminal  Sep 20 11:40 AM
hp@aditi: ~/Desktop/BTech/AUP/LAB/4

hp@aditi: ~/Desktop/BTech/TEST_CODE/aup-main/...  hp@aditi: ~/Desktop/BTech/AUP/LAB/4  hp@aditi: ~/Desktop/BTech/AUP/LAB/3

After a(buf), before b(buf) optimization
public_var = 1
static_var = 2
auto_var = 3

setjmp(buf) = 42
a_static_var = 104
a_auto_var = 5
public_var = 101
static_var = 102

Before a(buf)
public_var = 1
static_var = 2
auto_var = 3

a_static_var = 4
a_auto_var = 5
public_var = 1
static_var = 2

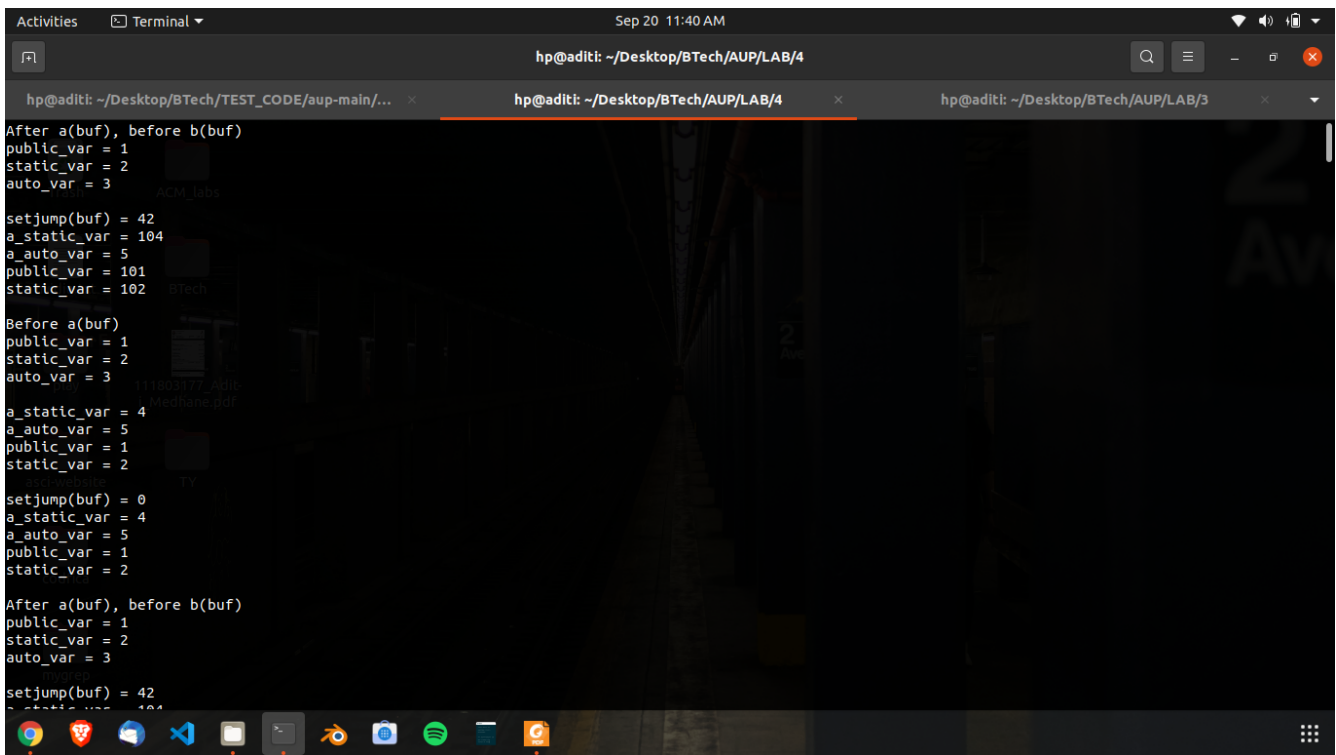
setjmp(buf) = 0
a_static_var = 4
a_auto_var = 5
public_var = 1
static_var = 2

After a(buf), before b(buf) redundant flags? ---This flag has no effect on the generated code, but it makes the
public_var = 1
static_var = 2
auto_var = 3

setjmp(buf) = 42
a_static_var = 104

```

Figure 3: Output for unoptimized code



```
hp@aditi: ~/Desktop/BTech/AUP/LAB/4
After a(buf), before b(buf)
public_var = 1
static_var = 2
auto_var = 3

setjump(buf) = 42
a_static_var = 104
a_auto_var = 5
public_var = 101
static_var = 102

Before a(buf)
public_var = 1
static_var = 2
auto_var = 3

a_static_var = 4
a_auto_var = 5
public_var = 1
static_var = 2

setjump(buf) = 0
a_static_var = 4
a_auto_var = 5
public_var = 1
static_var = 2

After a(buf), before b(buf)
public_var = 1
static_var = 2
auto_var = 3

setjump(buf) = 42
```

Figure 4: Output for optimized code

Q4

Creates a total of three different processes. each processes compute the factorial of integers between 1 and 10 by recursion and print the results to the screen and then terminate. Make sure to print an identifying string for the output of each process as in:

```
PROCESS1:fact(1)=1
PROCESS2:fact(2)=1
PROCESS2:fact(2)=2
PROCESS1:fact(2)=2
```

Code

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <pwd.h>
4  #include <sys/types.h>
5  #include <sys/stat.h>
6  #include <unistd.h>
7  #include <fcntl.h>
8  #include <errno.h>
9  #include <dirent.h>
10
11
12 #define FACT_NUM 10
13 #define NO_OF_CHILDREN 2
14
15 long int fact(int n){
16     long int f;
17     if( n == 1){
18         f = 1;
19     }
20     else{
21         f = n * fact(n - 1);
22     }
23     return f;
```

```

24
25 }
26
27 void work_done_by_child(int child_num){
28     int i;
29     //PRINT FACTORIAL VALUES FROM n = 1 to FACT_NUM
30     for(i = 1; i <= FACT_NUM; i++){
31         printf("Process%d:fact(%d)=%ld\n", child_num, i, fact(i));
32     }
33 }
34 }
35
36 int main(){
37     int child, i;
38     //STEP 1: CREATE CHILDREN
39     //STEP 2: DISPATCH EACH CHILD TP CALCULATE & PRINT FACT (DONE USING work_done_by_child)
40     for(i = 0; i < NO_OF_CHILDREN; i++){
41         if((child = fork()) == -1){
42             //FORK FAILED
43             perror("Creation of child process failed");
44         }
45         else if(child == 0){
46             work_done_by_child(i + 1);
47             return 0;
48         }
49     }
50
51     return 0;
52 }
53
54

```

Output

```

hp@aditi: ~/Desktop/BTech/AUP/LAB/4
hp@aditi: ~/Desktop/BTech/AUP/LAB/4
hp@aditi: ~/Desktop/BTech/AUP/LAB/4$ vim 4.c
hp@aditi: ~/Desktop/BTech/AUP/LAB/4$ gcc 4.c -o 4
hp@aditi: ~/Desktop/BTech/AUP/LAB/4$ ./4
Process2:fact(1)=1
Process2:fact(2)=2
Process2:fact(3)=6
Process1:fact(1)=1
Process1:fact(2)=2
Process1:fact(3)=6
Process1:fact(4)=24
Process1:fact(5)=120
Process1:fact(6)=720
Process1:fact(7)=5040
Process1:fact(8)=40320
Process1:fact(9)=362880
Process1:fact(10)=3628800
Process2:fact(4)=24
Process2:fact(5)=120
Process2:fact(6)=720
Process2:fact(7)=5040
Process2:fact(8)=40320
Process2:fact(9)=362880
Process2:fact(10)=3628800
hp@aditi: ~/Desktop/BTech/AUP/LAB/4$

```

Figure 5: PROCESS 1 & 2 fact calc & print