

AUP Assignment 5

111703013 Akshay Rajesh Deodhar

15th September 2020

Q1

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

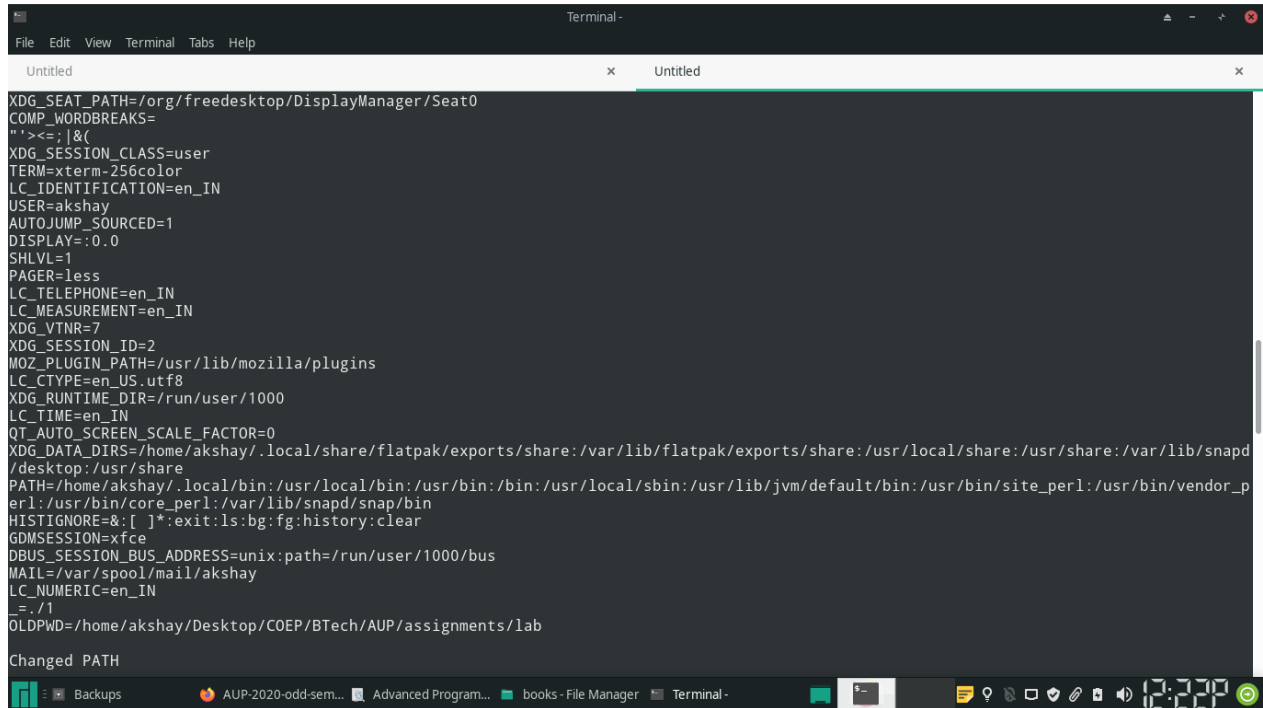
```
1
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include <errno.h>
5
6  void printenv(char **envp) {
7      while (*envp) {
8          printf("%s\n", *envp);
9          envp++;
10     }
11 }
12
13 int main(void) {
14     extern char **environ;
15
16     printenv(environ);
17
18     printf("\n");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

Answer

- Initially, the **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 its parent. But when a child changes the value of its environment, the value of *environ* for its parent does not change.
- Hence the output observed

Output



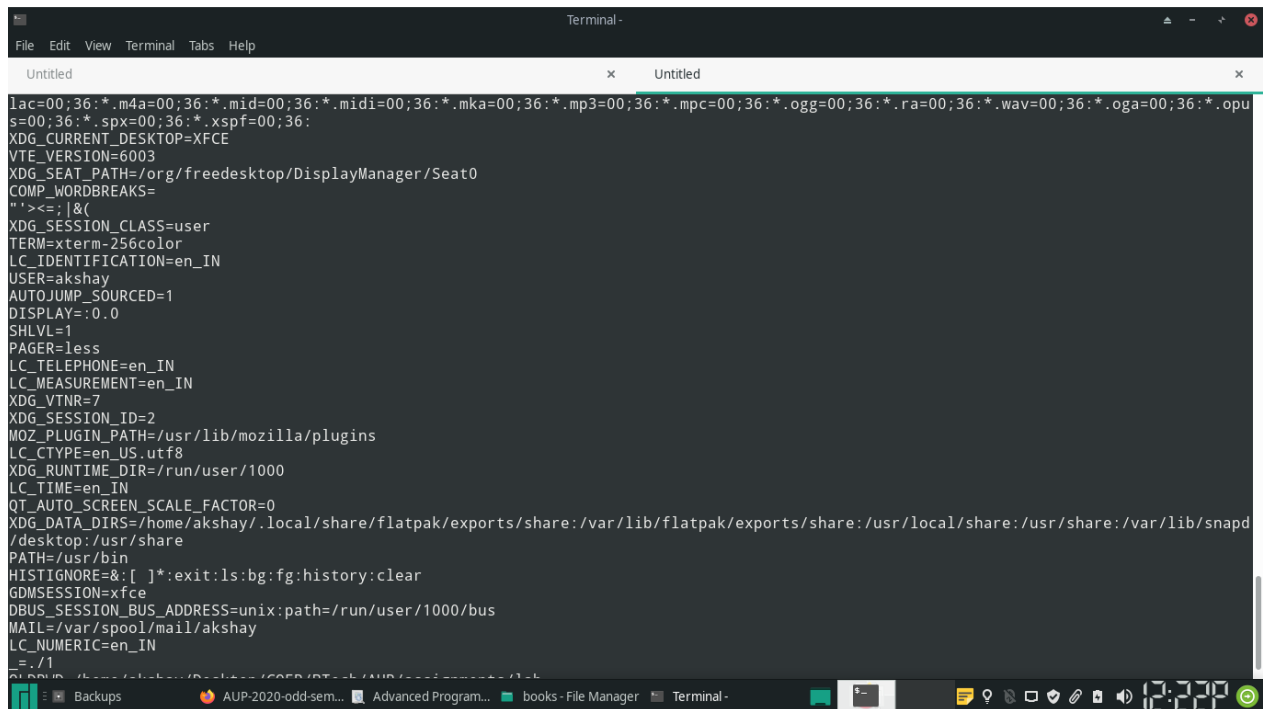
```

XDG_SEAT_PATH=/org/freedesktop/DisplayManager/Seat0
COMP_WORDBREAKS=
">=<=:|&{
XDG_SESSION_CLASS=user
TERM=xterm-256color
LC_IDENTIFICATION=en_IN
USER=akshay
AUTOJUMP_SOURCED=1
DISPLAY=:0.0
SHLVL=1
PAGER=less
LC_TELEPHONE=en_IN
LC_MEASUREMENT=en_IN
XDG_VTNR=7
XDG_SESSION_ID=2
MOZ_PLUGIN_PATH=/usr/lib/mozilla/plugins
LC_CTYPE=en_US.utf8
XDG_RUNTIME_DIR=/run/user/1000
LC_TIME=en_IN
QT_AUTO_SCREEN_SCALE_FACTOR=0
XDG_DATA_DIRS=/home/akshay/.local/share/flatpak/exports/share:/var/lib/flatpak/exports/share:/usr/local/share:/usr/share:/var/lib/snapd
/desktop:/usr/share
PATH=/home/akshay/.local/bin:/usr/local/bin:/usr/bin:/bin:/usr/local/sbin:/usr/lib/jvm/default/bin:/usr/bin/site_perl:/usr/bin/vendor_p
erl:/usr/bin/core_perl:/var/lib/snapd/snap/bin
HISTIGNORE=&[ ]*:exit:ls:bg:fg:history:clear
GDMSESSION=xfce
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
MAIL=/var/spool/mail/akshay
LC_NUMERIC=en_IN
_=/1
OLDPWD=/home/akshay/Desktop/COEP/BTech/AUP/assignments/lab

Changed PATH

```

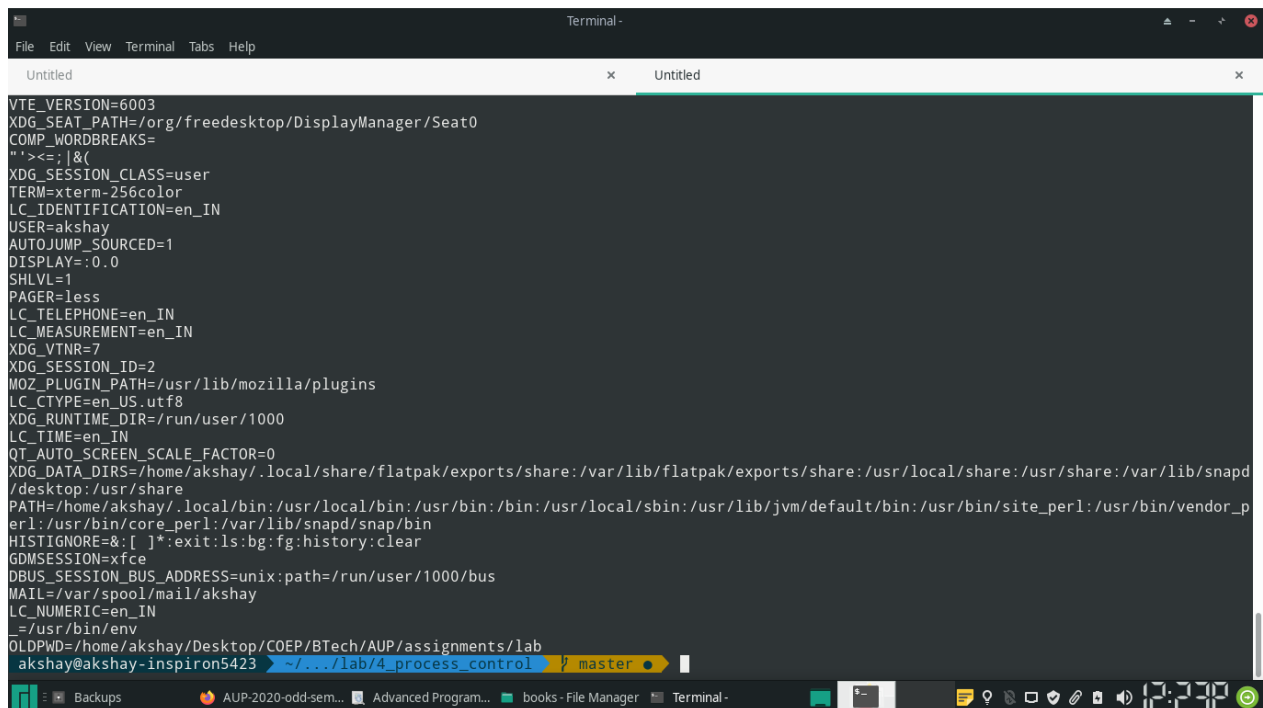
Figure 1: Value of PATH variable printed in program before it is changed



A terminal window titled "Terminal-" with a menu bar (File, Edit, View, Terminal, Tabs, Help) and two tabs labeled "Untitled". The terminal displays a list of environment variables. The PATH variable is highlighted in green, showing its updated value: `PATH=/usr/bin`. The variables include audio file formats, desktop settings, user information, and system paths.

```
lac=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:
XDG_CURRENT_DESKTOP=XFCE
VTE_VERSION=6003
XDG_SEAT_PATH=/org/freedesktop/DisplayManager/Seat0
COMP_WORDBREAKS=
"'"><=:|&({
XDG_SESSION_CLASS=user
TERM=xterm-256color
LC_IDENTIFICATION=en_IN
USER=akshay
AUTOJUMP_SOURCED=1
DISPLAY=:0.0
SHLVL=1
PAGER=less
LC_TELEPHONE=en_IN
LC_MEASUREMENT=en_IN
XDG_VTNR=7
XDG_SESSION_ID=2
MOZ_PLUGIN_PATH=/usr/lib/mozilla/plugins
LC_CTYPE=en_US.utf8
XDG_RUNTIME_DIR=/run/user/1000
LC_TIME=en_IN
QT_AUTO_SCREEN_SCALE_FACTOR=0
XDG_DATA_DIRS=/home/akshay/.local/share/flatpak/exports/share:/var/lib/flatpak/exports/share:/usr/local/share:/usr/share:/var/lib/snapd
/desktop:/usr/share
PATH=/usr/bin
HISTIGNORE=&[ ]*:exit:ls:bg:fg:history:clear
GDMSESSION=xfce
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
MAIL=/var/spool/mail/akshay
LC_NUMERIC=en_IN
_=./1
```

Figure 2: Value of PATH variable after it is changed using putenv



A terminal window titled "Terminal-" with a menu bar (File, Edit, View, Terminal, Tabs, Help) and two tabs labeled "Untitled". The terminal displays environment variables after a program has exited. The PATH variable is highlighted in green, showing its updated value: `PATH=/home/akshay/.local/bin:/usr/local/bin:/usr/bin:/bin:/usr/local/sbin:/usr/lib/jvm/default/bin:/usr/bin/site_perl:/usr/bin/vendor_perl:/usr/bin/core_perl:/var/lib/snapd/snap/bin`. The variables include desktop settings, user information, and system paths.

```
VTE_VERSION=6003
XDG_SEAT_PATH=/org/freedesktop/DisplayManager/Seat0
COMP_WORDBREAKS=
"'"><=:|&({
XDG_SESSION_CLASS=user
TERM=xterm-256color
LC_IDENTIFICATION=en_IN
USER=akshay
AUTOJUMP_SOURCED=1
DISPLAY=:0.0
SHLVL=1
PAGER=less
LC_TELEPHONE=en_IN
LC_MEASUREMENT=en_IN
XDG_VTNR=7
XDG_SESSION_ID=2
MOZ_PLUGIN_PATH=/usr/lib/mozilla/plugins
LC_CTYPE=en_US.utf8
XDG_RUNTIME_DIR=/run/user/1000
LC_TIME=en_IN
QT_AUTO_SCREEN_SCALE_FACTOR=0
XDG_DATA_DIRS=/home/akshay/.local/share/flatpak/exports/share:/var/lib/flatpak/exports/share:/usr/local/share:/usr/share:/var/lib/snapd
/desktop:/usr/share
PATH=/home/akshay/.local/bin:/usr/local/bin:/usr/bin:/bin:/usr/local/sbin:/usr/lib/jvm/default/bin:/usr/bin/site_perl:/usr/bin/vendor_p
erl:/usr/bin/core_perl:/var/lib/snapd/snap/bin
HISTIGNORE=&[ ]*:exit:ls:bg:fg:history:clear
GDMSESSION=xfce
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
MAIL=/var/spool/mail/akshay
LC_NUMERIC=en_IN
_=usr/bin/env
OLDPWD=/home/akshay/Desktop/COEP/BTech/AUP/assignments/lab
akshay@akshay-inspiron5423 ~/.../lab/4_process_control master
```

Figure 3: Value of PATH variable after program exits, for parent shell

Q2

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
2  #include <stdio.h>
3  #include <setjmp.h>
4  #include <stdlib.h>
5
6  int public_var;
7  jmp_buf buf;
8  static int static_var;
9
10 int a(jmp_buf buf) {
11     static int a_static_var;
12     auto int a_auto_var;
13     int ret;
14
15     a_static_var = 4;
16     a_auto_var = 5;
17
18     printf("a_static_var = %d\na_auto_var = %d\npublic_var = %d\nstatic_var = %d\n\n",
19           a_static_var, a_auto_var, public_var, static_var);
20
21     ret = setjmp(buf);
22
23     printf("setjump(buf) = %d\n", ret);
24     printf("a_static_var = %d\na_auto_var = %d\npublic_var = %d\nstatic_var = %d\n\n",
25           a_static_var, a_auto_var, public_var, static_var);
26
27     a_static_var = 104;
28     a_auto_var = 105;
29
30     return ret;
31 }
32
33 int b(jmp_buf buf) {
34     longjmp(buf, 42);
35 }
36
37
38 int main(void) {
```

```

39     auto int auto_var;
40
41     public_var = 1;
42     static_var = 2;
43     auto_var = 3;
44
45     printf("Before a(buf)\n");
46     printf("public_var = %d\nstatic_var = %d\nauto_var = %d\n\n",
47           public_var, static_var, auto_var);
48
49     if (a(buf)) {
50         exit(0);
51     }
52
53     printf("After a(buf), before b(buf)\n");
54     printf("public_var = %d\nstatic_var = %d\nauto_var = %d\n\n",
55           public_var, static_var, auto_var);
56
57     public_var = 101;
58     static_var = 102;
59     auto_var = 103;
60
61     b(buf);
62
63     printf("After b(buf)\n");
64     printf("public_var = %d\nstatic_var = %d\nauto_var = %d\n\n",
65           public_var, static_var, auto_var);
66
67     return 0;
68 }

```

Output

Answer

In case of the program above, the values of the static and global variables will *not* be rolled back to initial state. This is observed in the output.

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.

```
Terminal -
File Edit View Terminal Tabs Help
Untitled x Untitled x
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
a_static_var = 104
a_auto_var = 105
public_var = 101
static_var = 102

After b(buf)
public_var = 101
static_var = 102
auto_var = 103

akshay@akshay-inspiron5423 ~/.../lab/4_process_control master
aup/2.c at master · gaurav... [cfe-dev] One day left for ... Terminal - 23:53
```

Figure 4: Output for unoptimized code

```
Terminal -
File Edit View Terminal Tabs Help
Untitled x Untitled x
akshay@akshay-inspiron5423 ~/.../lab/4_process_control master ./2_optimized
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
a_static_var = 104
a_auto_var = 5
public_var = 101
static_var = 102

akshay@akshay-inspiron5423 ~/.../lab/4_process_control master
aup/2.c at master · gaurav... [cfe-dev] One day left for ... Terminal - 23:53
```

Figure 5: Output for optimized code

Q3

Creates a total of three different processes. Has each of the children 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 child process as in:

CHILD1:fact(1)=1

CHILD2:fact(2)=1

CHILD2:fact(2)=2

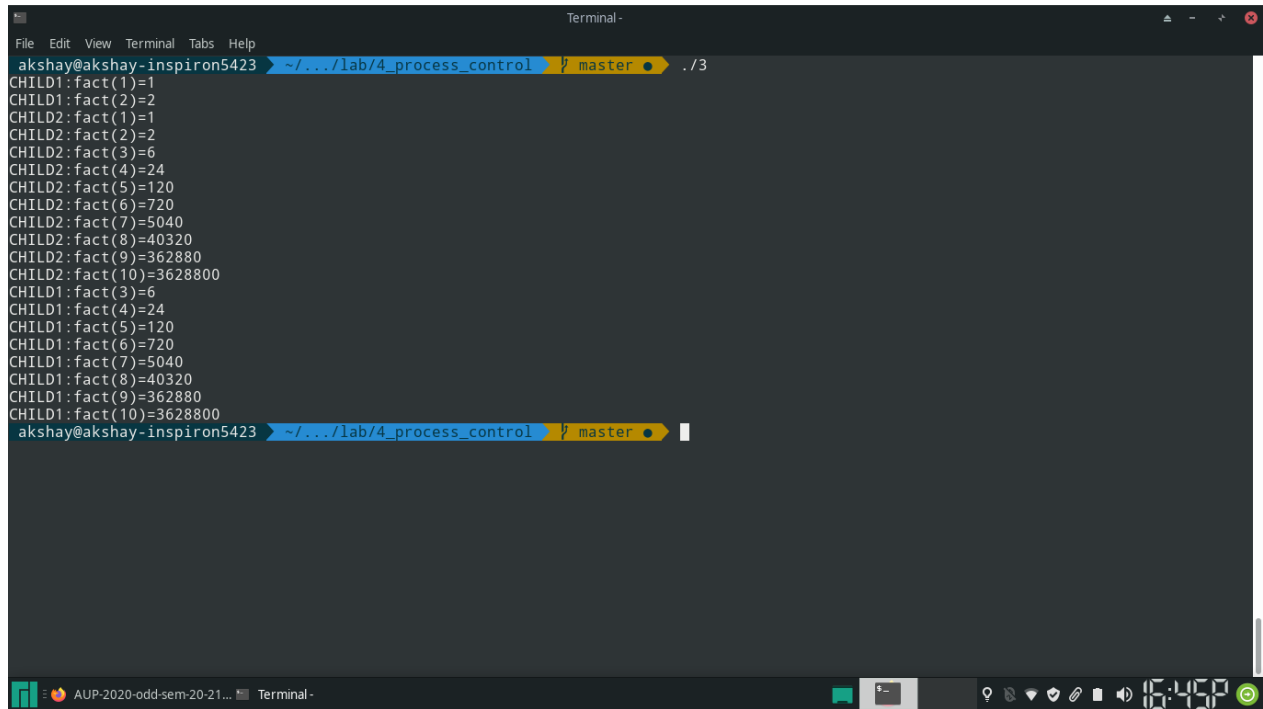
CHILD1:fact(2)=2

Code

```
1
2  #include <sys/types.h>
3  #include <unistd.h>
4  #include <stdio.h>
5
6  #define N_CHILDREN 2
7  #define F_N 10
8
9
10     /* calcualte factorial using recursion */
11     long int factorial(int n) {
12         long int f;
13         if (n == 1) {
14             f = 1;
15         }
16         else {
17             f = n * factorial(n - 1);
18         }
19         return f;
20     }
21
22 void child_work(int child_number) {
23     int i;
24     /* print factorial values from n = 1 to 10 */
25     for (i = 1; i <= F_N; i++) {
26         printf("CHILD%d:fact(%d)=%ld\n", child_number, i, factorial(i));
27     }
28 }
29
30 int main(void) {
31     int i;
32     int child;
33     /* create children, and dispatch each child to calcualte and print
34      * factorials */
35     for (i = 0; i < N_CHILDREN; i++) {
36         if ((child = fork()) == -1) {
37             /* fork failed */
38             perror("unable to create child process");
39         }
40         else if (child == 0) {
41             /* child */
42             child_work(i + 1);
43             return 0;
44         }
```

```
45     }  
46  
47     return 0;  
48 }
```

Output



A terminal window titled "Terminal -" showing the output of a program. The prompt is "akshay@akshay-inspiron5423" and the directory is "~/.../lab/4_process_control". The command executed is ". /3". The output shows two child processes, CHILD1 and CHILD2, calculating factorials from 1 to 10. CHILD2 prints its results first, followed by CHILD1. The results are: CHILD2: fact(1)=1, fact(2)=2, fact(3)=6, fact(4)=24, fact(5)=120, fact(6)=720, fact(7)=5040, fact(8)=40320, fact(9)=362880, fact(10)=3628800; CHILD1: fact(3)=6, fact(4)=24, fact(5)=120, fact(6)=720, fact(7)=5040, fact(8)=40320, fact(9)=362880, fact(10)=3628800. The terminal window has a menu bar with File, Edit, View, Terminal, Tabs, and Help. The bottom status bar shows the system clock as 16:45 and various system icons.

```
akshay@akshay-inspiron5423 ~/.../lab/4_process_control master ● ./3  
CHILD1: fact(1)=1  
CHILD1: fact(2)=2  
CHILD2: fact(1)=1  
CHILD2: fact(2)=2  
CHILD2: fact(3)=6  
CHILD2: fact(4)=24  
CHILD2: fact(5)=120  
CHILD2: fact(6)=720  
CHILD2: fact(7)=5040  
CHILD2: fact(8)=40320  
CHILD2: fact(9)=362880  
CHILD2: fact(10)=3628800  
CHILD1: fact(3)=6  
CHILD1: fact(4)=24  
CHILD1: fact(5)=120  
CHILD1: fact(6)=720  
CHILD1: fact(7)=5040  
CHILD1: fact(8)=40320  
CHILD1: fact(9)=362880  
CHILD1: fact(10)=3628800  
akshay@akshay-inspiron5423 ~/.../lab/4_process_control master ●
```

Figure 6: Both processes calculating and printing factorial