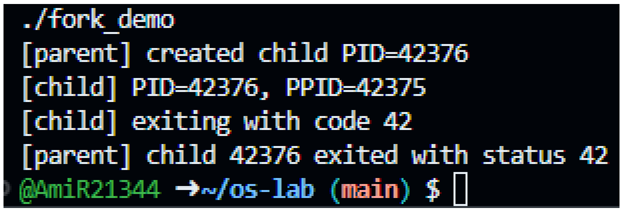
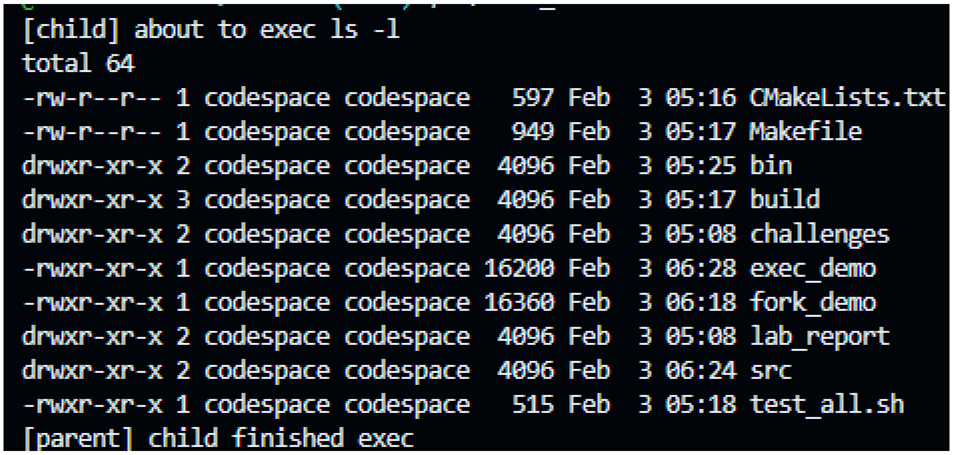
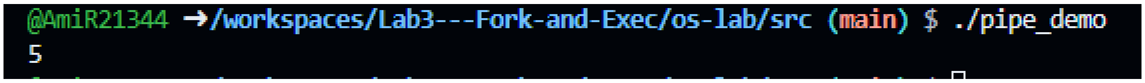


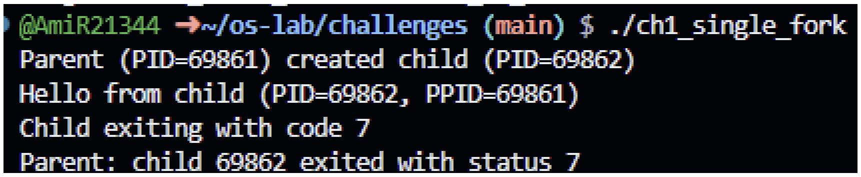
```
./fork_demo
[parent] created child PID=42376
[child] PID=42376, PPID=42375
[child] exiting with code 42
[parent] child 42376 exited with status 42
@AmiR21344 ➜~/os-lab (main) $ 
```



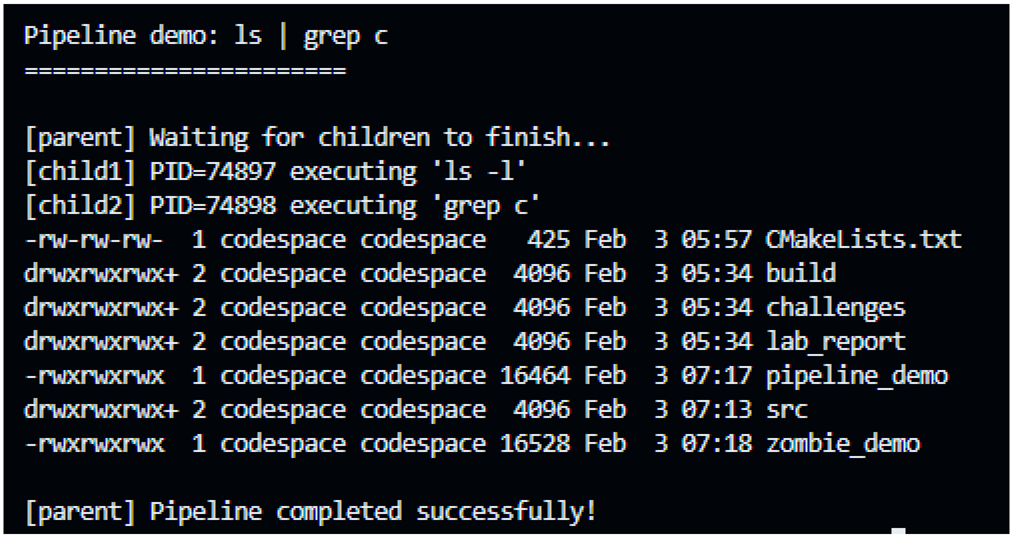
```
[child] about to exec ls -l
total 64
-rw-r--r-- 1 codespace codespace   597 Feb  3 05:16 CMakeLists.txt
-rw-r--r-- 1 codespace codespace   949 Feb  3 05:17 Makefile
drwxr-xr-x 2 codespace codespace  4096 Feb  3 05:25 bin
drwxr-xr-x 3 codespace codespace  4096 Feb  3 05:17 build
drwxr-xr-x 2 codespace codespace  4096 Feb  3 05:08 challenges
-rwxr-xr-x 1 codespace codespace 16200 Feb  3 06:28 exec_demo
-rwxr-xr-x 1 codespace codespace 16360 Feb  3 06:18 fork_demo
drwxr-xr-x 2 codespace codespace  4096 Feb  3 05:08 lab_report
drwxr-xr-x 2 codespace codespace  4096 Feb  3 06:24 src
-rwxr-xr-x 1 codespace codespace   515 Feb  3 05:18 test_all.sh
[parent] child finished exec
```



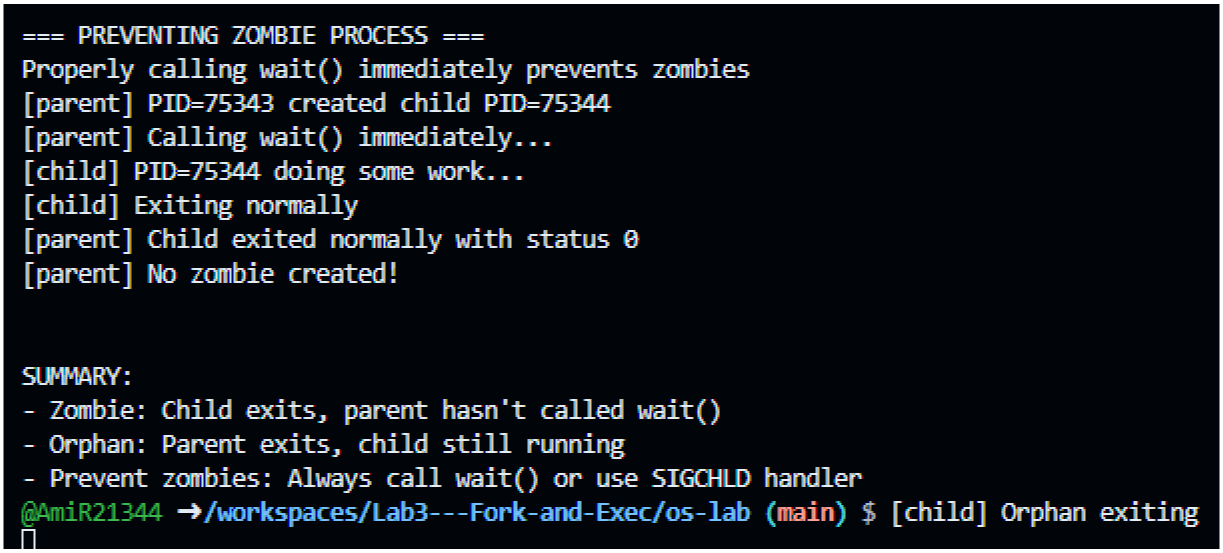
```
@AmiR21344 ➜/workspaces/Lab3---Fork-and-Exec/os-lab/src (main) $ ./pipe_demo
5
```



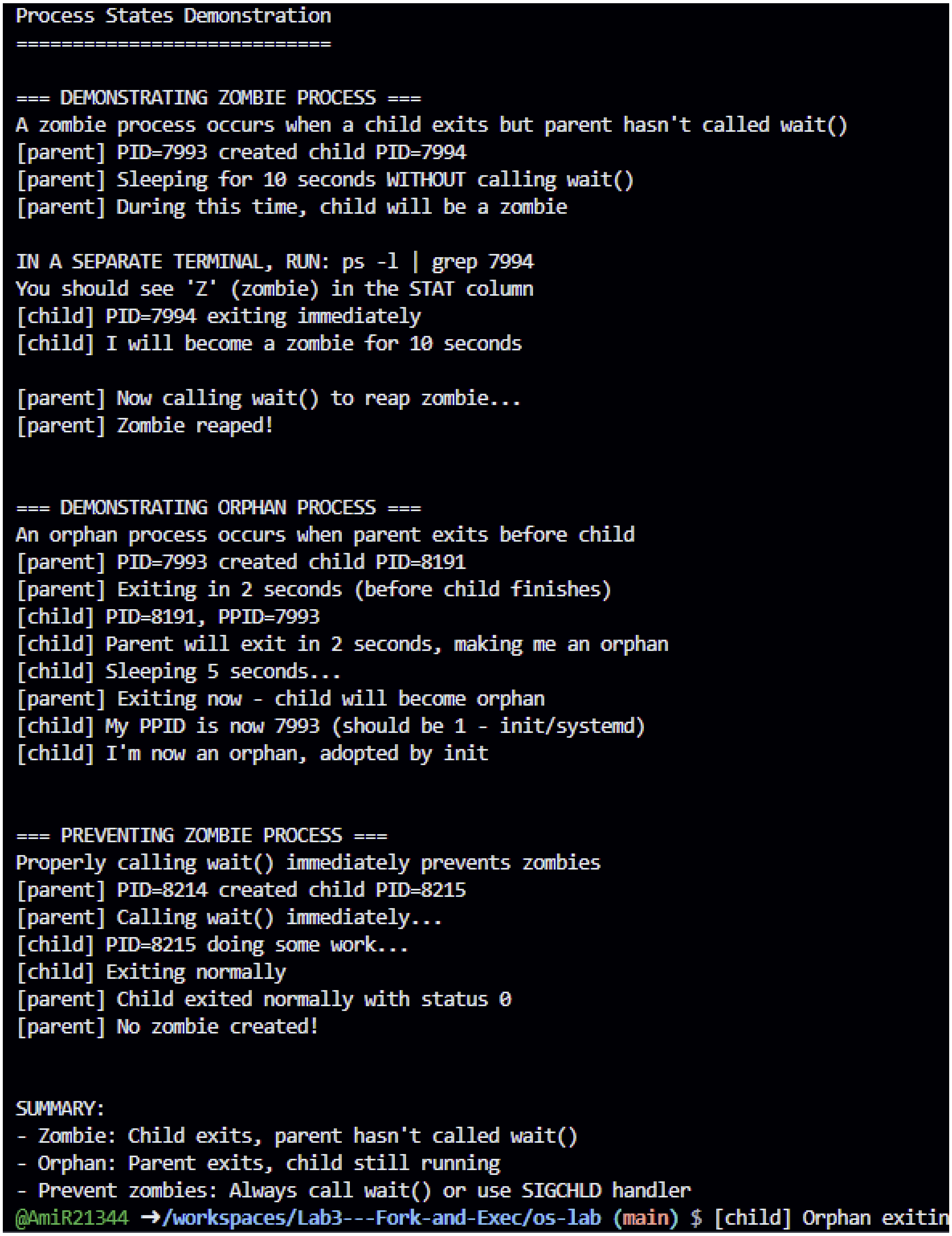
```
@AmiR21344 ➜~/os-lab/challenges (main) $ ./ch1_single_fork
Parent (PID=69861) created child (PID=69862)
Hello from child (PID=69862, PPID=69861)
Child exiting with code 7
Parent: child 69862 exited with status 7
```



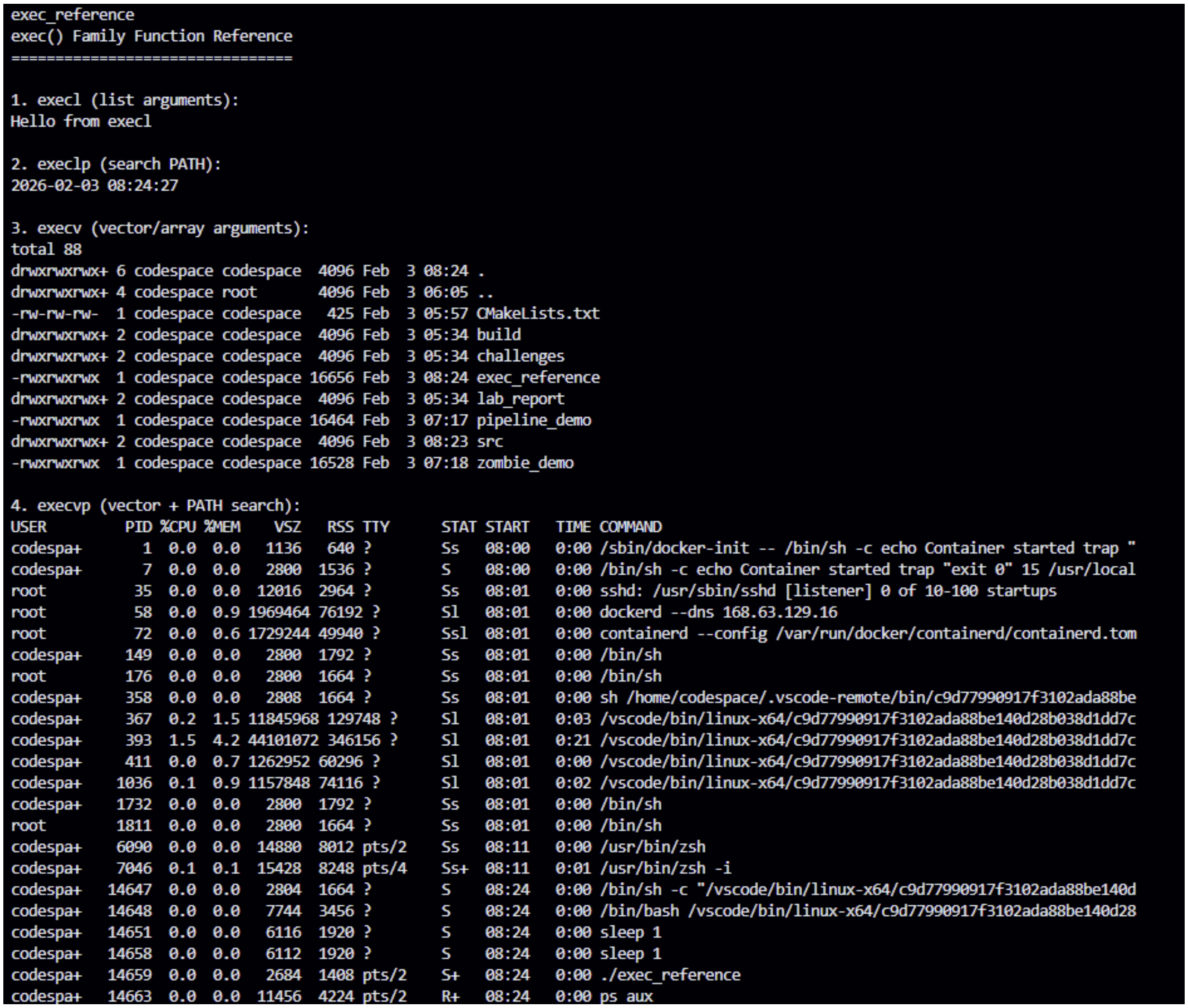
```
Pipeline demo: ls | grep c
=======================

[parent] Waiting for children to finish...
[child1] PID=74897 executing 'ls -l'
[child2] PID=74898 executing 'grep c'
-rw-rw-rw-  1 codespace codespace   425 Feb  3 05:57 CMakeLists.txt
drwxrwxrwx+ 2 codespace codespace  4096 Feb  3 05:34 build
drwxrwxrwx+ 2 codespace codespace  4096 Feb  3 05:34 challenges
drwxrwxrwx+ 2 codespace codespace  4096 Feb  3 05:34 lab_report
-rwxrwxrwx  1 codespace codespace 16464 Feb  3 07:17 pipeline_demo
drwxrwxrwx+ 2 codespace codespace  4096 Feb  3 07:13 src
-rwxrwxrwx  1 codespace codespace 16528 Feb  3 07:18 zombie_demo

[parent] Pipeline completed successfully!
```



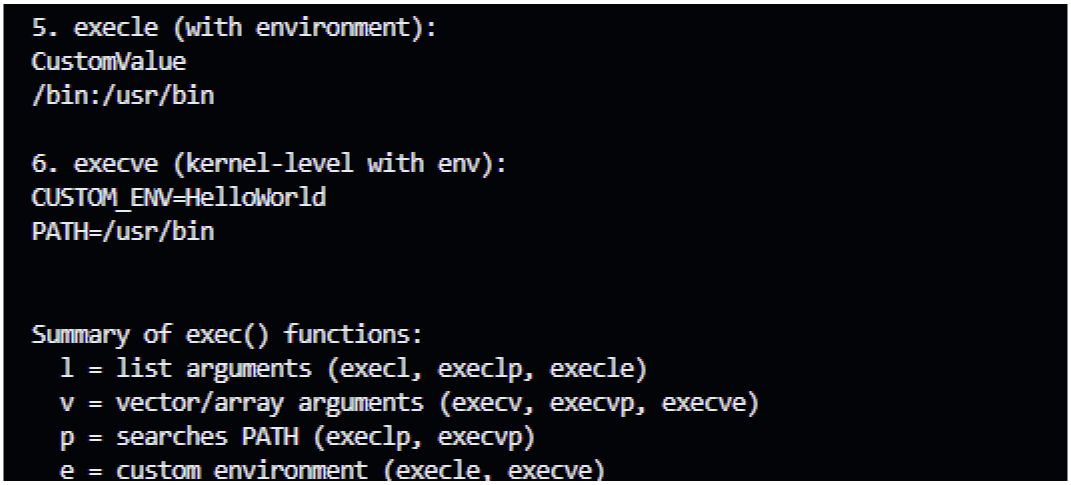
```
=== PREVENTING ZOMBIE PROCESS ===
Properly calling wait() immediately prevents zombies
[parent] PID=75343 created child PID=75344
[parent] Calling wait() immediately...
[child] PID=75344 doing some work...
[child] Exiting normally
[parent] Child exited normally with status 0
[parent] No zombie created!


SUMMARY:
- Zombie: Child exits, parent hasn't called wait()
- Orphan: Parent exits, child still running
- Prevent zombies: Always call wait() or use SIGCHLD handler
@AmiR21344 ➜ /workspaces/Lab3---Fork-and-Exec/os-lab (main) $ [child] Orphan exiting
```



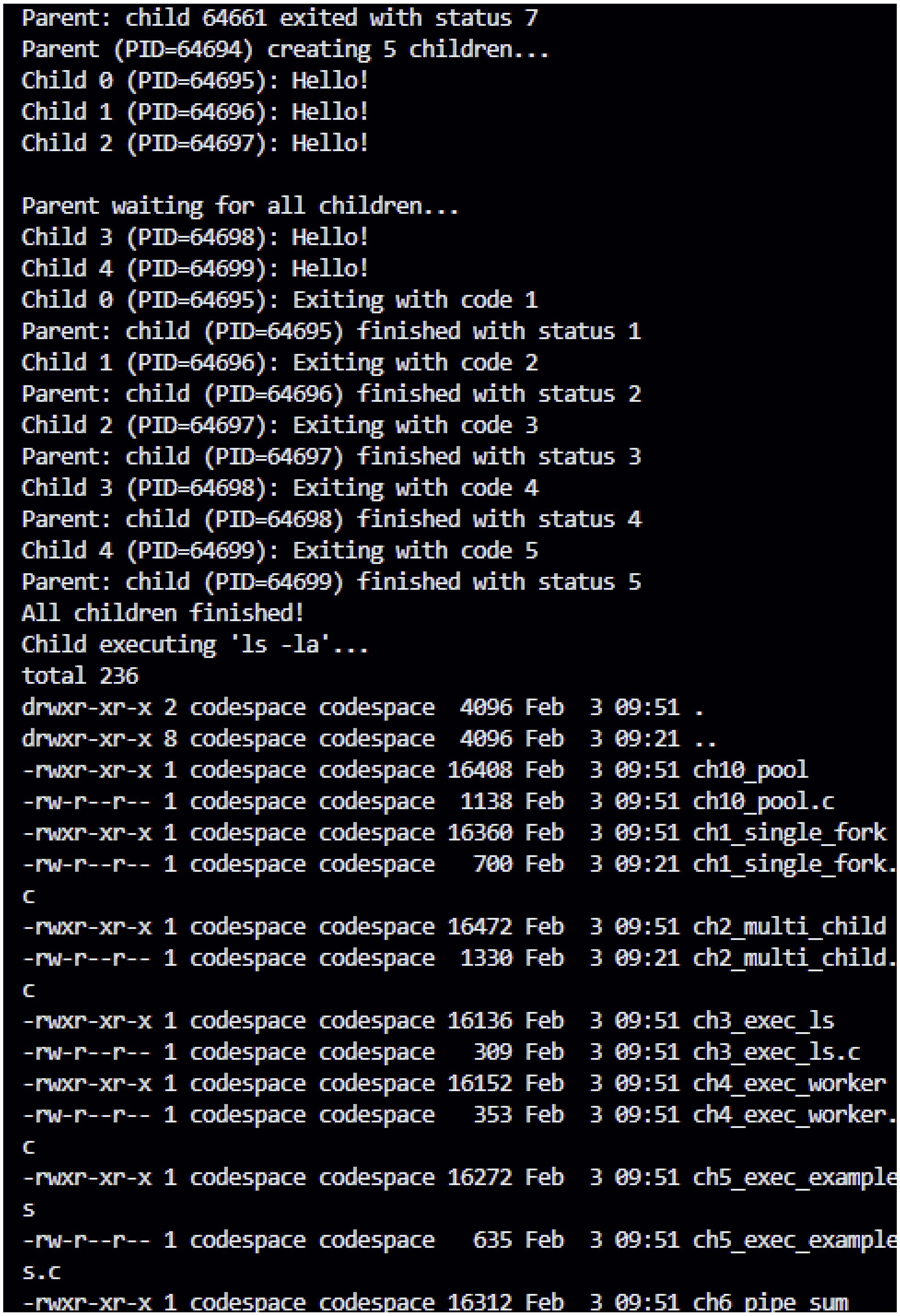
```
Process States Demonstration
=============================

=== DEMONSTRATING ZOMBIE PROCESS ===
A zombie process occurs when a child exits but parent hasn't called wait()
[parent] PID=7993 created child PID=7994
[parent] Sleeping for 10 seconds WITHOUT calling wait()
[parent] During this time, child will be a zombie

IN A SEPARATE TERMINAL, RUN: ps -l | grep 7994
You should see 'Z' (zombie) in the STAT column
[child] PID=7994 exiting immediately
[child] I will become a zombie for 10 seconds

[parent] Now calling wait() to reap zombie...
[parent] Zombie reaped!


=== DEMONSTRATING ORPHAN PROCESS ===
An orphan process occurs when parent exits before child
[parent] PID=7993 created child PID=8191
[parent] Exiting in 2 seconds (before child finishes)
[child] PID=8191, PPID=7993
[child] Parent will exit in 2 seconds, making me an orphan
[child] Sleeping 5 seconds...
[parent] Exiting now - child will become orphan
[child] My PPID is now 7993 (should be 1 - init/systemd)
[child] I'm now an orphan, adopted by init


=== PREVENTING ZOMBIE PROCESS ===
Properly calling wait() immediately prevents zombies
[parent] PID=8214 created child PID=8215
[parent] Calling wait() immediately...
[child] PID=8215 doing some work...
[child] Exiting normally
[parent] Child exited normally with status 0
[parent] No zombie created!


SUMMARY:
- Zombie: Child exits, parent hasn't called wait()
- Orphan: Parent exits, child still running
- Prevent zombies: Always call wait() or use SIGCHLD handler
@AmiR21344 ➜/workspaces/Lab3---Fork-and-Exec/os-lab (main) $ [child] Orphan exitin
```



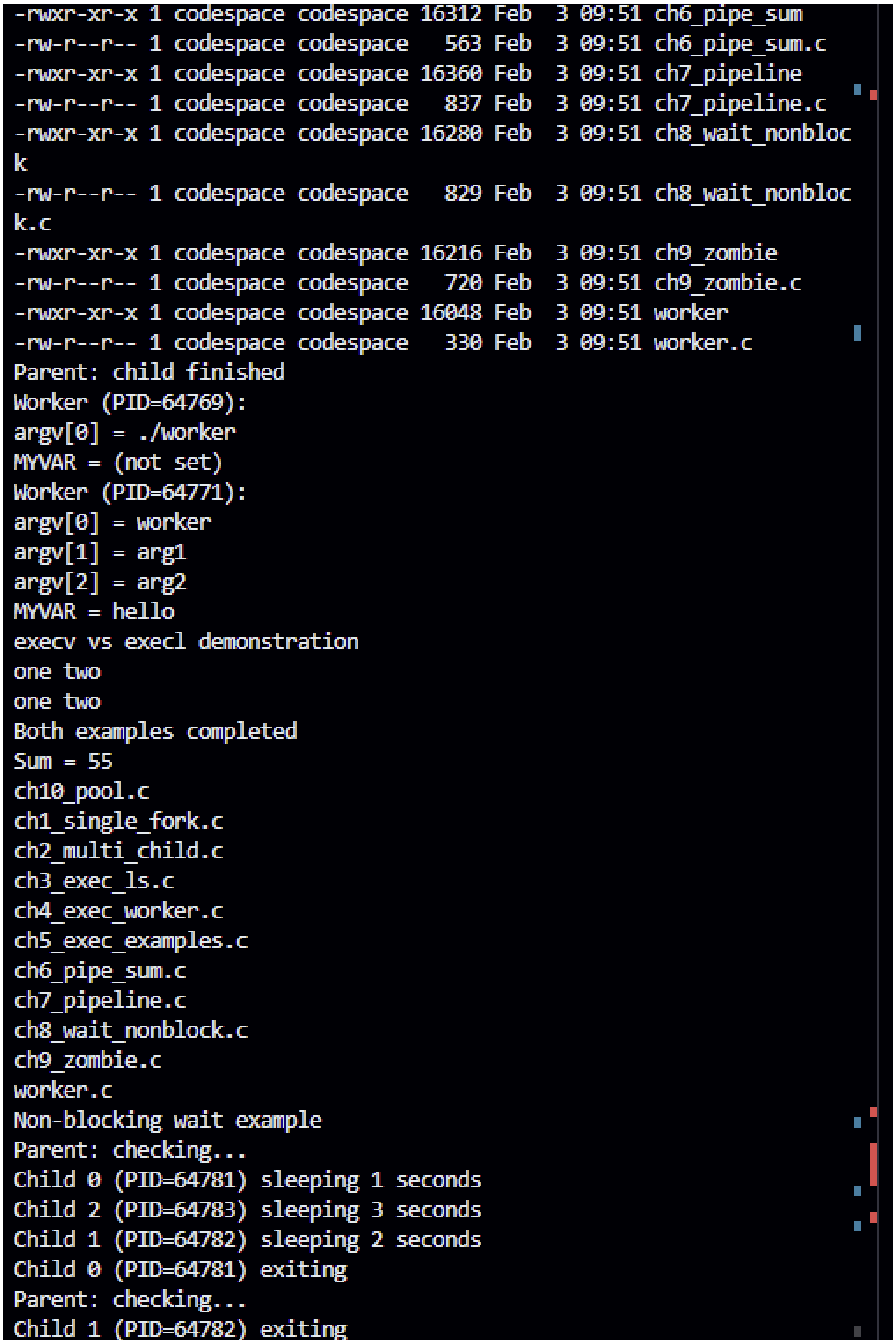
```
exec_reference
exec() Family Function Reference
================================

1. execl (list arguments):
Hello from execl

2. execlp (search PATH):
2026-02-03 08:24:27

3. execv (vector/array arguments):
total 88
drwxrwxrwx+ 6 codespace codespace  4096 Feb  3 08:24 .
drwxrwxrwx+ 4 codespace root       4096 Feb  3 06:05 ..
-rw-rw-rw-  1 codespace codespace   425 Feb  3 05:57 CMakeLists.txt
drwxrwxrwx+ 2 codespace codespace  4096 Feb  3 05:34 build
drwxrwxrwx+ 2 codespace codespace  4096 Feb  3 05:34 challenges
-rwxrwxrwx  1 codespace codespace 16656 Feb  3 08:24 exec_reference
drwxrwxrwx+ 2 codespace codespace  4096 Feb  3 05:34 lab_report
-rwxrwxrwx  1 codespace codespace 16464 Feb  3 07:17 pipeline_demo
drwxrwxrwx+ 2 codespace codespace  4096 Feb  3 08:23 src
-rwxrwxrwx  1 codespace codespace 16528 Feb  3 07:18 zombie_demo

4. execvp (vector + PATH search):
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
codespa+       1  0.0  0.0   1136   640 ?        Ss   08:00   0:00 /sbin/docker-init -- /bin/sh -c echo Container started trap "
codespa+       7  0.0  0.0   2800  1536 ?        S    08:00   0:00 /bin/sh -c echo Container started trap "exit 0" 15 /usr/local
root          35  0.0  0.0  12016  2964 ?        Ss   08:01   0:00 sshd: /usr/sbin/sshd [listener] 0 of 10-100 startups
root          58  0.0  0.9 1969464 76192 ?       Sl   08:01   0:00 dockerd --dns 168.63.129.16
root          72  0.0  0.6 1729244 49940 ?       Ssl  08:01   0:00 containerd --config /var/run/docker/containerd/containerd.tom
codespa+     149  0.0  0.0   2800  1792 ?        Ss   08:01   0:00 /bin/sh
root         176  0.0  0.0   2800  1664 ?        Ss   08:01   0:00 /bin/sh
codespa+     358  0.0  0.0   2808  1664 ?        Ss   08:01   0:00 sh /home/codespace/.vscode-remote/bin/c9d77990917f3102ada88be
codespa+     367  0.2  1.5 11845968 129748 ?     Sl   08:01   0:03 /vscode/bin/linux-x64/c9d77990917f3102ada88be140d28b038d1dd7c
codespa+     393  1.5  4.2 44101072 346156 ?     Sl   08:01   0:21 /vscode/bin/linux-x64/c9d77990917f3102ada88be140d28b038d1dd7c
codespa+     411  0.0  0.7 1262952 60296 ?       Sl   08:01   0:00 /vscode/bin/linux-x64/c9d77990917f3102ada88be140d28b038d1dd7c
codespa+    1036  0.1  0.9 1157848 74116 ?       Sl   08:01   0:02 /vscode/bin/linux-x64/c9d77990917f3102ada88be140d28b038d1dd7c
codespa+    1732  0.0  0.0   2800  1792 ?        Ss   08:01   0:00 /bin/sh
root        1811  0.0  0.0   2800  1664 ?        Ss   08:01   0:00 /bin/sh
codespa+    6090  0.0  0.0  14880  8012 pts/2    Ss   08:11   0:00 /usr/bin/zsh
codespa+    7046  0.1  0.1  15428  8248 pts/4    Ss+  08:11   0:01 /usr/bin/zsh -i
codespa+   14647  0.0  0.0   2804  1664 ?        S    08:24   0:00 /bin/sh -c "/vscode/bin/linux-x64/c9d77990917f3102ada88be140d
codespa+   14648  0.0  0.0   7744  3456 ?        S    08:24   0:00 /bin/bash /vscode/bin/linux-x64/c9d77990917f3102ada88be140d28
codespa+   14651  0.0  0.0   6116  1920 ?        S    08:24   0:00 sleep 1
codespa+   14658  0.0  0.0   6112  1920 ?        S    08:24   0:00 sleep 1
codespa+   14659  0.0  0.0   2684  1408 pts/2    S+   08:24   0:00 ./exec_reference
codespa+   14663  0.0  0.0  11456  4224 pts/2    R+   08:24   0:00 ps aux
```



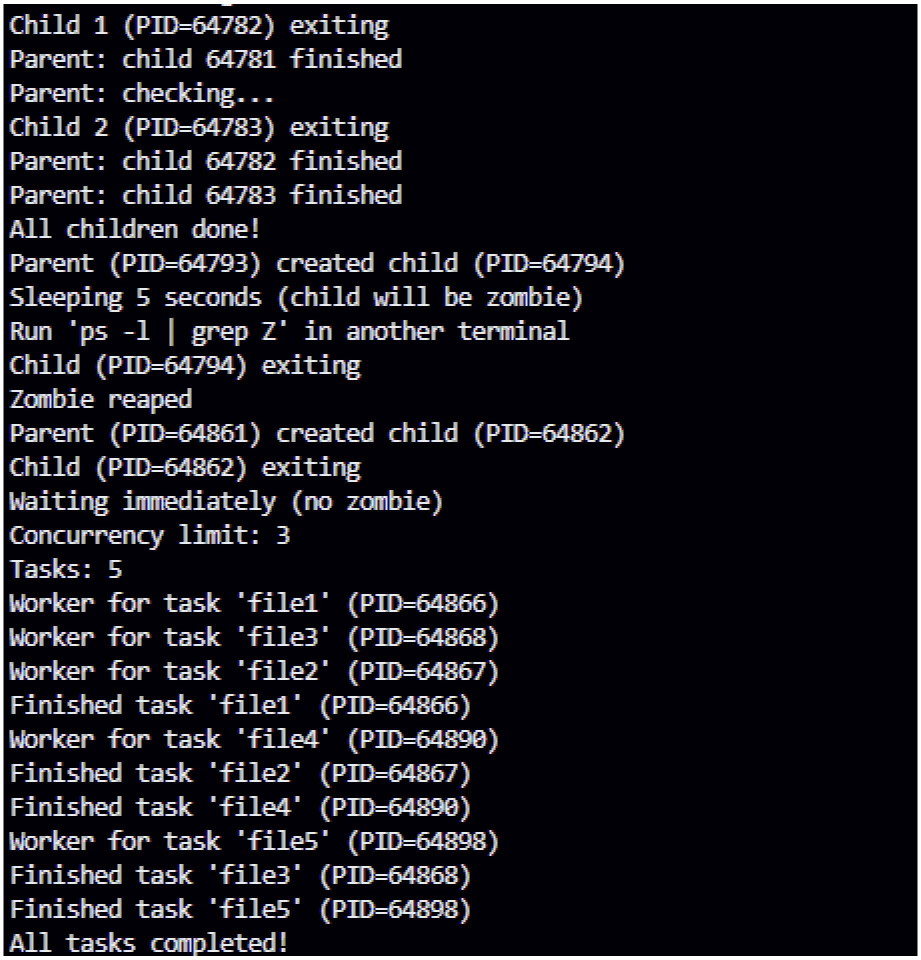
```
5. execle (with environment):
CustomValue
/bin:/usr/bin

6. execve (kernel-level with env):
CUSTOM_ENV=HelloWorld
PATH=/usr/bin


Summary of exec() functions:
  l = list arguments (execl, execlp, execle)
  v = vector/array arguments (execv, execvp, execve)
  p = searches PATH (execlp, execvp)
  e = custom environment (execle, execve)
```



```
Parent: child 64661 exited with status 7
Parent (PID=64694) creating 5 children...
Child 0 (PID=64695): Hello!
Child 1 (PID=64696): Hello!
Child 2 (PID=64697): Hello!

Parent waiting for all children...
Child 3 (PID=64698): Hello!
Child 4 (PID=64699): Hello!
Child 0 (PID=64695): Exiting with code 1
Parent: child (PID=64695) finished with status 1
Child 1 (PID=64696): Exiting with code 2
Parent: child (PID=64696) finished with status 2
Child 2 (PID=64697): Exiting with code 3
Parent: child (PID=64697) finished with status 3
Child 3 (PID=64698): Exiting with code 4
Parent: child (PID=64698) finished with status 4
Child 4 (PID=64699): Exiting with code 5
Parent: child (PID=64699) finished with status 5
All children finished!
Child executing 'ls -la'...
total 236
drwxr-xr-x 2 codespace codespace  4096 Feb  3 09:51 .
drwxr-xr-x 8 codespace codespace  4096 Feb  3 09:21 ..
-rwxr-xr-x 1 codespace codespace 16408 Feb  3 09:51 ch10_pool
-rw-r--r-- 1 codespace codespace  1138 Feb  3 09:51 ch10_pool.c
-rwxr-xr-x 1 codespace codespace 16360 Feb  3 09:51 ch1_single_fork
-rw-r--r-- 1 codespace codespace   700 Feb  3 09:21 ch1_single_fork.
c
-rwxr-xr-x 1 codespace codespace 16472 Feb  3 09:51 ch2_multi_child
-rw-r--r-- 1 codespace codespace  1330 Feb  3 09:21 ch2_multi_child.
c
-rwxr-xr-x 1 codespace codespace 16136 Feb  3 09:51 ch3_exec_ls
-rw-r--r-- 1 codespace codespace   309 Feb  3 09:51 ch3_exec_ls.c
-rwxr-xr-x 1 codespace codespace 16152 Feb  3 09:51 ch4_exec_worker
-rw-r--r-- 1 codespace codespace   353 Feb  3 09:51 ch4_exec_worker.
c
-rwxr-xr-x 1 codespace codespace 16272 Feb  3 09:51 ch5_exec_example
s
-rw-r--r-- 1 codespace codespace   635 Feb  3 09:51 ch5_exec_example
s.c
-rwxr-xr-x 1 codespace codespace 16312 Feb  3 09:51 ch6_pipe_sum
```



```
-rwxr-xr-x 1 codespace codespace 16312 Feb  3 09:51 ch6_pipe_sum
-rw-r--r-- 1 codespace codespace   563 Feb  3 09:51 ch6_pipe_sum.c
-rwxr-xr-x 1 codespace codespace 16360 Feb  3 09:51 ch7_pipeline
-rw-r--r-- 1 codespace codespace   837 Feb  3 09:51 ch7_pipeline.c
-rwxr-xr-x 1 codespace codespace 16280 Feb  3 09:51 ch8_wait_nonbloc
k
-rw-r--r-- 1 codespace codespace   829 Feb  3 09:51 ch8_wait_nonbloc
k.c
-rwxr-xr-x 1 codespace codespace 16216 Feb  3 09:51 ch9_zombie
-rw-r--r-- 1 codespace codespace   720 Feb  3 09:51 ch9_zombie.c
-rwxr-xr-x 1 codespace codespace 16048 Feb  3 09:51 worker
-rw-r--r-- 1 codespace codespace   330 Feb  3 09:51 worker.c
Parent: child finished
Worker (PID=64769):
argv[0] = ./worker
MYVAR = (not set)
Worker (PID=64771):
argv[0] = worker
argv[1] = arg1
argv[2] = arg2
MYVAR = hello
execv vs execl demonstration
one two
one two
Both examples completed
Sum = 55
ch10_pool.c
ch1_single_fork.c
ch2_multi_child.c
ch3_exec_ls.c
ch4_exec_worker.c
ch5_exec_examples.c
ch6_pipe_sum.c
ch7_pipeline.c
ch8_wait_nonblock.c
ch9_zombie.c
worker.c
Non-blocking wait example
Parent: checking...
Child 0 (PID=64781) sleeping 1 seconds
Child 2 (PID=64783) sleeping 3 seconds
Child 1 (PID=64782) sleeping 2 seconds
Child 0 (PID=64781) exiting
Parent: checking...
Child 1 (PID=64782) exiting
```



```
Child 1 (PID=64782) exiting
Parent: child 64781 finished
Parent: checking...
Child 2 (PID=64783) exiting
Parent: child 64782 finished
Parent: child 64783 finished
All children done!
Parent (PID=64793) created child (PID=64794)
Sleeping 5 seconds (child will be zombie)
Run 'ps -l | grep Z' in another terminal
Child (PID=64794) exiting
Zombie reaped
Parent (PID=64861) created child (PID=64862)
Child (PID=64862) exiting
Waiting immediately (no zombie)
Concurrency limit: 3
Tasks: 5
Worker for task 'file1' (PID=64866)
Worker for task 'file3' (PID=64868)
Worker for task 'file2' (PID=64867)
Finished task 'file1' (PID=64866)
Worker for task 'file4' (PID=64890)
Finished task 'file2' (PID=64867)
Finished task 'file4' (PID=64890)
Worker for task 'file5' (PID=64898)
Finished task 'file3' (PID=64868)
Finished task 'file5' (PID=64898)
All tasks completed!
```