

Linux-Process-API-fork-wait-exec Public

forked from gowriganeshns/Linux-Process-API-fork-wait-exec

1 Branch 0 Tags Go to file Add file Code

This branch is 1 commit ahead of gowriganeshns/Linux-Process-API-fork-wait-exec:main .

Contribute Sync fork

ganeshprabhu2005 Update README.md 0c6cd72 · now

LICENSE	Initial commit	9 months ago
README.md	Update README.md	now
execcheck2.c	Create execcheck2.c	8 months ago
forkcheck.c	Create forkcheck.c	8 months ago
pidcheck.c	Create pidcheck.c	8 months ago

README License

# Linux-Process-API-fork-wait-exec-

Ex02-Linux Process API-fork(), wait(), exec()

## Ex02-OS-Linux-Process API - fork(), wait(), exec()

Operating systems Lab exercise

Ex02-Linux Process API-fork(), wait(), exec()

- Readme
- GPL-3.0 license
- Activity
- 0 stars
- 0 watching
- 0 forks

### Releases

No releases published Create a new release

### Packages

No packages published Publish your first package

### Languages

- C 100.0%

### Suggested workflows

Based on your tech stack

CMake based, multi-platform projects Configure Build and test a CMake based project on multiple platforms.

# AIM:

To write C Program that uses Linux Process API - fork(), wait(), exec()

## DESIGN STEPS:

### Step 1:

Navigate to any Linux environment installed on the system or installed inside a virtual environment like virtual box/vmware or online linux JSLinux (<https://bellard.org/jslinux/vm.html?url=alpine-x86.cfg&mem=192>) or docker.

### Step 2:

Write the C Program using Linux Process API - fork(), wait(), exec()

### Step 3:

Test the C Program for the desired output.

## PROGRAM:

### C Program to print process ID and parent Process ID using Linux API system calls

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main(void)
{
    //variable to store calling function's process
    id
    pid_t process_id;
    //variable to store parent function's process
    id
    pid_t p_process_id;
    //getpid() - will return process id of calling
    function
    process_id = getpid();
    //getppid() - will return process id of parent
    function
    p_process_id = getppid();
    //printing the process ids

    //printing the process ids
```



**MSBuild based projects**

Configure

Build a MSBuild based project.



**SLSA Generic generator**

Configure

Generate SLSA3 provenance for your existing release workflows

[More workflows](#)

[Dismiss suggestions](#)

```

        printf("The process id: %d\n",process_id);
        printf("The process id of parent function:
%d\n",p_process_id);
        return 0; }

```

## ##OUTPUT

```

vishwaraj@vishwaraj-VirtualBox:~$ ./execcheck
Running ps with execlp
child exited with status of 0
Done.
Running ps with execlp. Now with path specified
  PID TTY          STAT TIME   COMMAND
    1 ?        Ss   0:11 /lib/systemd/systemd --system --deserialize 45 spla
    2 ?        S    0:00 [kthreadd]
    3 ?        I<   0:00 [rcu_gp]
    4 ?        I<   0:00 [rcu_par_gp]
    5 ?        I<   0:00 [slub_flushwq]
    6 ?        I<   0:00 [netns]
    8 ?        I<   0:00 [kworker/0:0H-events_highpri]
   11 ?        I<   0:00 [mre_percpu_wq]
   12 ?        I    0:00 [rcu_tasks_kthread]
   13 ?        I    0:00 [rcu_tasks_rude_kthread]
   14 ?        I    0:00 [rcu_tasks_trace_kthread]
   15 ?        S    0:02 [ksoftirqd/0]
   16 ?        I    1:11 [rcu_preempt]
   17 ?        S    0:00 [migration/0]
   18 ?        S    0:00 [idle_inject/0]
   19 ?        S    0:00 [cpuhp/0]
   20 ?        S    0:00 [cpuhp/1]
   21 ?        S    0:00 [idle_inject/1]
   22 ?        S    0:00 [migration/1]
   23 ?        S    0:21 [ksoftirqd/1]
   26 ?        S    0:00 [kdevtmpfs]
   27 ?        I<   0:00 [inet_frag_wq]
   29 ?        S    0:00 [kauditd]
   30 ?        S    0:00 [khungtaskd]
   31 ?        S    0:00 [oom_reaper]
   32 ?        I<   0:00 [writeback]
   33 ?        S    0:00 [kcompactd0]
   34 ?        SN   0:00 [ksmd]
   35 ?        SN   0:00 [khugepaged]
   36 ?        I<   0:00 [kintegrityd]
   37 ?        I<   0:00 [ablockd]
   38 ?        I<   0:00 [blkcg_punt_bio]
   39 ?        I<   0:00 [tpm_dev_wq]
   40 ?        I<   0:00 [ata_sff]

```

## C Program to create new process using Linux API system calls fork() and exit()

```

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main() {
    int pid;
    pid = fork();
    if (pid == -1) {
        perror("fork");
        exit(EXIT_FAILURE);
    }
    else if (pid == 0) {
        printf("I am child, my pid is %d\n",
getpid());
        printf("My parent pid is: %d\n", getppid());
        exit(EXIT_SUCCESS);
    }
    else {
        printf("I am parent, my pid is %d\n",
getpid());
        sleep(100);
        exit(EXIT_SUCCESS);
    }
    return 0;
}

```



```

22350 ? Ss 0:00 avahi-daemon: running [vishwaraj-VirtualBox.local]
22353 ? S 0:00 avahi-daemon: chroot helper
41890 ? Ss 0:38 /lib/systemd/systemd-oomd
44712 ? I 0:05 [kworker/u4:1-events_unbound]
48701 ? I< 0:03 [kworker/1:2H-kblockd]
48868 ? Sl 0:00 /usr/libexec/gvfsd-network --spawner :1.3 /org/gtk/
48886 ? Sl 0:00 /usr/libexec/gvfsd-dnssd --spawner :1.3 /org/gtk/gv
49835 ? Ssl 0:04 /usr/libexec/fwupd/fwupd
51834 ? I< 0:01 [kworker/0:1H-kblockd]
55027 ? I 0:00 [kworker/1:1-events]
55153 ? Rsl 0:29 /usr/libexec/gnome-terminal-server
55171 pts/0 Ss 0:00 bash
55181 ? Sl 3:40 /snap/firefox/2987/usr/lib/firefox/firefox
55320 ? Sl 0:00 /snap/firefox/2987/usr/lib/firefox/firefox -content
55341 ? Sl 0:31 /snap/firefox/2987/usr/lib/firefox/firefox -content
55501 ? Sl 0:00 /snap/firefox/2987/usr/lib/firefox/firefox -content
55797 ? Sl 1:07 /snap/firefox/2987/usr/lib/firefox/firefox -content
55914 ? I 0:01 [kworker/0:2-events]
56032 ? Sl 0:06 gjs /usr/share/gnome-shell/extensions/ding@rasterso
56086 ? I 0:03 [kworker/1:0-events]
56089 ? I 0:01 [kworker/u4:2-events_unbound]
56194 ? I 0:00 [kworker/0:3-events]
56249 ? Ssl 0:22 /snap/code/152/usr/share/code/code --no-sandbox --f
56251 ? S 0:00 /snap/code/152/usr/share/code/code --type=zygote --
56252 ? S 0:00 /snap/code/152/usr/share/code/code --type=zygote --
56265 ? Sl 0:00 /snap/code/152/usr/share/code/chrome_crashpad_handl
56292 ? Sl 0:00 /snap/code/152/usr/share/code/code --type=utility -
56320 ? Sl 1:47 /snap/code/152/usr/share/code/code --type=renderer
56358 ? Sl 0:35 /snap/code/152/usr/share/code/code --type=gpu-proce
56383 ? Sl 0:08 /snap/code/152/usr/share/code/code --type=utility -
56478 ? Sl 0:02 /snap/code/152/usr/share/code/code --type=utility -
56489 ? Sl 0:13 /snap/code/152/usr/share/code/code --type=utility -
56535 ? Sl 0:02 /snap/code/152/usr/share/code/code /snap/code/152/u
56557 ? Sl 0:00 /snap/firefox/2987/usr/lib/firefox/firefox -content
56579 ? Sl 0:00 /snap/firefox/2987/usr/lib/firefox/firefox -content
56606 ? Sl 0:00 /snap/firefox/2987/usr/lib/firefox/firefox -content
56885 ? I 0:01 [kworker/u4:3-events_power_efficient]
57042 ? Sl 0:05 /snap/code/152/usr/share/code/code --type=renderer
57313 pts/0 R+ 0:00 ps ax
vishwaraj@vishwaraj-VirtualBox: $

```

## C Program to execute Linux system commands using Linux API system calls exec() family

```

#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>

int main() {
    pid_t pid = fork();
    if (pid < 0) {
        perror("Fork failed");
        exit(EXIT_FAILURE);
    } else if (pid == 0) {
        printf("This is the child process. Executing
'ls' command.\n");
        execl("/bin/ls", "ls", "-l", NULL); // Lists
files in long format
        perror("execl failed");
        exit(EXIT_FAILURE);
    } else {
        int status;
        waitpid(pid, &status, 0); // Wait for the
child to finish
        if (WIFEXITED(status)) {
            printf("Child process exited with status
%d.\n", WEXITSTATUS(status));
        } else {
            printf("Child process did not exit
normally.\n");
        }
        printf("Parent process is done.\n");
    }
}

```



```
    return 0;  
}
```

##OUTPUT

## RESULT:

---

The programs are executed successfully.