به نام خدا

آزمایشگاه سیستم عامل

process overview &

FORK System call

in Linux

نویسنده:

علیرضا اخوان پور

www.AlirezaWeb.com

Info@alirezaweb.com

چهارشنبه ۷ اردیبهشت ماه ، ۱۳۹۲

مشخصه های پروسه:

در لینوکس هر پروسه خصیصه ای دارد که میتوان با دستور ps آن را مشاهده کرد؛ در زیر بعضی از این خصیصه ها را توضیح خواهم داد:

USER: کاربری که پروسس به آن تعلق دارد.

شناسه‌ی پروسس یا PID: شماره یکتایی است که برای ارجاء پروسس از آن استفاده می‌شود.

شناسه پروسس پدر یا PPID : شماره پروسسی که این پروسه را اجرا کرده.

TTY: ترمینالی که پروسس به آن متصل است.

```
alireza@alireza-Inspiron-N5010:~$ ps -ux
warning: bad ps syntax, perhaps a bogus '-'?
See http://gitorious.org/procps/procps/blobs/master/Documentation/FAQ
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
alireza   1888  0.0  0.0   6984   776 tty1     S    23:46   0:00 /bin/login --
alireza   1993  0.1  0.0   6616  3024 tty1     S+   23:46   0:00 -bash
alireza   2143  0.0  0.1  66180  4256 ?        Sl   23:47   0:00 /usr/bin/gnome-
alireza   2154  0.6  0.2  52088  9672 ?        Ssl  23:47   0:00 gnome-session -
alireza   2192  0.0  0.0   3844   528 ?        S    23:47   0:00 dbus-launch --a
alireza   2193  0.0  0.0   4096   204 ?        Ss   23:47   0:00 /usr/bin/ssh-ag
alireza   2196  0.0  0.0   3844   528 ?        S    23:47   0:00 /usr/bin/dbus-l
alireza   2197  1.3  0.0   5328  2236 ?        Ss   23:47   0:00 //bin/dbus-daem
alireza   2198  0.0  0.0   3248   392 ?        Ss   23:47   0:00 //bin/dbus-daem
alireza   2200  0.0  0.0  46480  3912 ?        Sl   23:47   0:00 /usr/lib/at-spi
alireza   2204  0.0  0.0   3380  1420 ?        S    23:47   0:00 /bin/dbus-daemo
alireza   2207  0.0  0.0  17092  3060 ?        Sl   23:47   0:00 /usr/lib/at-spi
alireza   2216  1.6  0.4 176904 17376 ?        Sl   23:47   0:00 /usr/lib/gnome-
alireza   2226  0.1  0.0  26880  2828 ?        Sl   23:47   0:00 /usr/lib/gvfs/g
alireza   2230  0.0  0.0  45172  3604 ?        Sl   23:47   0:00 /usr/lib/gvfs//
alireza   2238  9.0  1.8 268248 74608 ?        Sl   23:47   0:02 compiz
alireza   2247  0.0  0.0   3724   784 ?        S    23:47   0:00 syndaemon -i 2.
alireza   2254  0.3  0.1 108884  5288 ?        S<l  23:47   0:00 /usr/bin/pulsea
alireza   2256  0.2  0.0  33832  2592 ?        Sl   23:47   0:00 /usr/lib/dconf/
alireza   2259  2.9  0.5 147268 23372 ?        Sl   23:47   0:00 nautilus -n
```

همان طور که در بالا اشاره کردیم, PID هر پروسس منحصر به فرد و UNIC است, و با ارسال سیگنال های کنترلی مختلف به هر پروسس می توانیم آن را مدیریت کنیم.

برای مثال اگر بخواهیم پروسسی را Kill کنیم باید سیگنال شماره ۱۵ را به آن ارسال کنیم.

Kill –p PID

یا

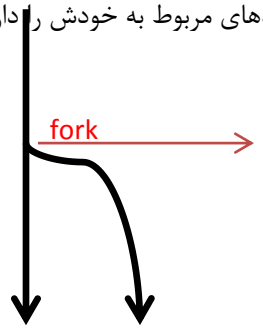
killall myprogramm

*** نکته:** در واقع وقتی در حین اجرای یک پروسس که در ترمینال پیش رویمان قرار دارد CTRL+C را می‌زنیم, سیستم عامل همین سیگنال شماره ۱۵ را به آن پروسس می فرستد و با kill کردن آن, آن پروسس را متوقف می‌سازد.

Fork یا چنگال:

این مفهوم در سیستم عامل‌های مختلف برقرار است,

هر جا در کدهای برنامه‌یمان دستور fork را به کار می‌بریم, از آن به بعد آن خطوط توسط ۲ پروسس هم زمان اجرا می‌شوند و داده‌های مورد نیاز پروسس فرزند از پروسس پدر در آن کپی شده و از آن پس هر پروسس داده‌های مربوط به خودش را دارند.



هنگامی که دستور fork فراخوانی می‌شود؛
یک پروسس دیگر در سیستم عامل ایجاد می‌شود
که به این پروسس, پروسس فرزند و پروسس قبلی
پدر نامیده می‌شود.

پس از fork, در صورتی که عملیات ایجاد پروسس جدید موفقیت آمیز باشد,

اگر ما id را بررسی کنیم, با 0 بودن آن در می‌یابیم که در پروسه‌ی فرزندیم, در غیر این صورت در پروسه‌ی پدر هستیم.

```
childPID = fork();

if(childPID >= 0) // متوجه می‌شویم عمل ایجاد پروسه‌ی فرزند موفقیت آمیز بوده.

{

  if(childPID == 0)     {

      فرآیند فرزند

  }

  else

  {

      فرآیند پدر

  }

}
```

بررسی فرآیند و ایجاد پروسس فرزند در سیستم عامل لینوکس با استفاده از زبان C++:

در برنامه نویسی هر جا از system call ـِ fork استفاده کنیم, می توانید با بررسی صحت شرط **pid>=0** از صحت ایجاد پروسه‌ی جدید فرزند مطلع شویم,
در غیر این صورت اگر این مقدار ۱- باشد, عملیات با شکست مواجه شده.

- هر جا system call ـِ **()getpid** را در برنامه‌یمان اجرا کنیم, PID یا پروسس آی‌دی **همین برنامه** را می‌دهد.
- هر جا system call ـِ **()getppid** را در برنامه‌یمان اجرا کنیم, PID یا پروسس آی‌دی **پدر** را می‌دهد.

پروسس parent به همه‌ی childهایی که ایجاد می‌کند اشراف کامل دارد, مثلاً میتواند اگر فرزندی resource اضافه مصرف می‌کند, از آن بگیرد یا آن را kill کند یا ... .

دستور waite :

معطل می‌ماند تا اگر پروسس childی دارد, آن پروسس تمام شده و exitشه, بعد دستورات دیگر را اجرا می‌کند.

دستور sleep :

برحسب ثانیه پروسس جاری را متوقف می‌کند؛
برای مثال sleep(1) به مدت ۱ ثانیه منتظر می‌ماند, سپس بقیه‌ی دستورات را اجرا می‌کند.

مثال: محیط زیر, ویرایش‌گر vim می‌باشد, در زیر از system callهای ذکر شده در بالا استفاده شده...

```c
/* Includes */
#include <unistd.h>     /* Symbolic Constants */
#include <sys/types.h>  /* Primitive System Data Types */
#include <errno.h>      /* Errors */
#include <stdio.h>      /* Input/Output */
#include <sys/wait.h>   /* Wait for Process Termination */
#include <stdlib.h>     /* General Utilities */

int main()
{
    pid_t childpid; /* variable to store the child's pid */
    int retval;     /* child process: user-provided return code */
    int status;     /* parent process: child's exit status */

    /* only 1 int variable is needed because each process would have its
       own instance of the variable
       here, 2 int variables are used for clarity */

    /* now create new process */
    childpid = fork();

    if (childpid >= 0) /* fork succeeded */
    {
        if (childpid == 0) /* fork() returns 0 to the child process */
        {
            printf("CHILD: I am the child process!\n");
            printf("CHILD: Here's my PID: %d\n", getpid());
            printf("CHILD: My parent's PID is: %d\n", getppid());
            printf("CHILD: The value of my copy of childpid is: %d\n", childpid);
            printf("CHILD: Sleeping for 1 second...\n");
            sleep(1); /* sleep for 1 second */
            printf("CHILD: Enter an exit value (0 to 255): ");
            scanf(" %d", &retval);
            printf("CHILD: Goodbye!\n");
            exit(retval); /* child exits with user-provided return code */
        }
        else /* fork() returns new pid to the parent process */
        {
            printf("PARENT: I am the parent process!\n");
            printf("PARENT: Here's my PID: %d\n", getpid());
            printf("PARENT: The value of my copy of childpid is %d\n", childpid);
            printf("PARENT: I will now wait for my child to exit.\n");
            wait(&status); /* wait for child to exit, and store its status */
            printf("PARENT: Child's exit code is: %d\n", WEXITSTATUS(status));
            printf("PARENT: Goodbye!\n");
            exit(0);  /* parent exits */
        }
    }
    else /* fork returns -1 on failure */
    {
        perror("fork"); /* display error message */
        exit(0);
    }
}
```

و اجرای این برنامه:

```
alireza@alireza-Inspiron-N5010:~/fork$ vim fork-ex.c
alireza@alireza-Inspiron-N5010:~/fork$ g++ fork-ex.c
alireza@alireza-Inspiron-N5010:~/fork$ ./a.out
PARENT: I am the parent process!
PARENT: Here's my PID: 3443
PARENT: The value of my copy of childpid is 3444
PARENT: I will now wait for my child to exit.
CHILD: I am the child process!
CHILD: Here's my PID: 3444
CHILD: My parent's PID is: 3443
CHILD: The value of my copy of childpid is: 0
CHILD: Sleeping for 1 second...
CHILD: Enter an exit value (0 to 255): 15
CHILD: Goodbye!
PARENT: Child's exit code is: 15
PARENT: Goodbye!
alireza@alireza-Inspiron-N5010:~/fork$
```

از آنجایی که کلیه‌ی دستورات بعد fork در هر دو پروسه‌ی پدر و فرزند کپی می‌شود, ما با

نکته‌ی قابل توجه این است که بعد از دستور fork و ایجاد یک پروسس جدید, تضمینی نداریم که سیستم عامل به سراغ پروسه‌ی پدر می‌رود یا فرزند؟!

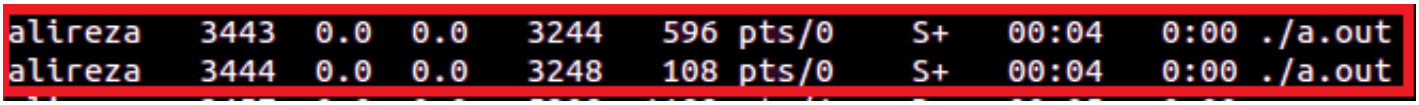
بسته به سیاست زمانبندی سیستم عامل, یکی از پروسه‌ها زود تر اجرا می‌شود و هر وقت سیستم عامل تصمیم بگیرد, می‌تواند موقتاً منابع را از آن پروسه بگیرد و با در اختار گذاشتن آن منابع به پروسه‌ی دیگر, پروسه‌ی دیگر را اجرا کند.

از آنجایی که ما هیچ مدیریتی روی context switch های سیستم عامل نداریم, نباید در برنامه نویسی حالت خاصی را برای ترتیب اجرای فرآیندهایمان در نظر بگیریم,

در نمونه‌ی بالا, **در این اجرای خاص**, اولین خطی که پس از fork اجرا شده, کدی از پروسسParent بوده.

حال, در حین اجرای برنامه‌یمان, دستور PS را در ترمینالی دیگر اجرا کردیم و تصویر زیر مشاهده شد:

```
alireza   2338  0.0  0.2  42816 10764 ?        Sl   Apr28   0:00 /usr/bin/gtk-window-decorator
alireza   2347  0.6  0.4  97396 19816 ?        Sl   Apr28   0:06 /usr/lib/unity/unity-panel-service
alireza   2349  0.3  0.1  58496  6208 ?        Sl   Apr28   0:03 /usr/lib/indicator-appmenu/hud-service
alireza   2358  0.0  0.1  55276  4288 ?        Sl   Apr28   0:00 /usr/lib/i386-linux-gnu/indicator-application-service
alireza   2362  0.0  0.2  63060  9928 ?        Sl   Apr28   0:00 /usr/lib/indicator-printers/indicator-printers-service
alireza   2365  0.0  0.1  68400  5300 ?        Sl   Apr28   0:00 /usr/lib/indicator-session/indicator-session-service
alireza   2366  0.0  0.2  79020  8680 ?        Sl   Apr28   0:00 /usr/lib/indicator-datetime/indicator-datetime-service
alireza   2367  0.0  0.1  68056  4804 ?        Sl   Apr28   0:00 /usr/lib/indicator-messages/indicator-messages-service
alireza   2368  0.0  0.1 129952  6284 ?        Sl   Apr28   0:00 /usr/lib/indicator-sound/indicator-sound-service
alireza   2393  0.0  0.1  49820  6332 ?        Sl   Apr28   0:00 /usr/lib/evolution/evolution-source-registry
alireza   2405  0.0  0.1  32272  5308 ?        Sl   Apr28   0:00 /usr/lib/geoclue/geoclue-master
alireza   2421  0.0  0.1  42120  6280 ?        Sl   Apr28   0:00 /usr/lib/ubuntu-geoip/ubuntu-geoip-provider
alireza   2426  0.0  0.2  76040 10452 ?        Sl   Apr28   0:00 telepathy-indicator
alireza   2432  0.0  0.1  45108  7084 ?        Sl   Apr28   0:00 /usr/lib/telepathy/mission-control-5
alireza   2438  0.0  0.4 110036 19044 ?        Sl   Apr28   0:00 /usr/bin/signon-ui
alireza   2451  0.0  0.2  76260 10280 ?        Sl   Apr28   0:00 /usr/lib/unity-lens-applications/unity-applications-daemon
alireza   2453  0.0  0.1  70308  6540 ?        Sl   Apr28   0:00 /usr/lib/unity-lens-files/unity-files-daemon
alireza   2455  0.0  0.1  92312  5888 ?        Sl   Apr28   0:00 /usr/lib/gwibber/unity-gwibber-daemon
alireza   2457  0.0  0.1  82152  7620 ?        Sl   Apr28   0:00 /usr/lib/i386-linux-gnu/unity-music-daemon
alireza   2459  0.0  0.4  98116 19840 ?        Sl   Apr28   0:00 /usr/bin/python3 /usr/lib/unity-lens-photos/unity-lens-photos
alireza   2461  0.0  0.2  95968  8708 ?        Sl   Apr28   0:00 /usr/lib/i386-linux-gnu/unity-shopping-daemon
alireza   2463  0.0  0.3  86592 12580 ?        Sl   Apr28   0:00 /usr/bin/python /usr/lib/unity-lens-video/unity-lens-video
alireza   2489  0.0  0.1  46416  5264 ?        Sl   Apr28   0:00 /usr/bin/zeitgeist-daemon
alireza   2508  0.0  0.1  55876  5200 ?        Sl   Apr28   0:00 zeitgeist-datahub
alireza   2517  0.0  0.2  52944  9476 ?        Sl   Apr28   0:00 /usr/lib/zeitgeist/zeitgeist-fts
alireza   2527  0.0  0.0   4228   284 ?        S    Apr28   0:00 /bin/cat
alireza   2533  0.0  0.3 110068 15100 ?        Sl   Apr28   0:00 /usr/bin/python3 /usr/lib/unity-lens-files/unity-scope-gdocs
alireza   2535  0.0  0.1  89264  4484 ?        Sl   Apr28   0:00 /usr/lib/i386-linux-gnu/unity-musicstore-daemon
alireza   2581  0.0  0.3 103028 14060 ?        Sl   Apr28   0:00 /usr/bin/python /usr/lib/unity-scope-video-remote/unity-scope-video-remote
alireza   2651  0.0  0.3  52880 12260 ?        Sl   Apr28   0:00 update-notifier
alireza   2679  0.0  0.0  17312  2440 ?        Sl   Apr28   0:00 /usr/lib/gvfs/gvfsd-metadata
alireza   2686  0.0  0.1  37548  4044 ?        Sl   Apr28   0:00 /usr/lib/i386-linux-gnu/deja-dup/deja-dup-monitor
alireza   3097  0.0  0.1  88148  6100 ?        Sl   Apr28   0:00 /usr/lib/gvfs/gvfsd-http --spawner :1.7 /org/gtk/gvfs/exec_spaw/3
alireza   3255  0.5  0.4  95480 17512 ?        Sl   Apr28   0:02 gnome-terminal
alireza   3260  0.0  0.0   2404   740 ?        S    Apr28   0:00 gnome-pty-helper
alireza   3261  0.0  0.0   6232  2740 pts/0    Ss   Apr28   0:00 bash
alireza   3383  0.0  0.0   6172  2548 pts/1    Ss   00:04   0:00 bash
alireza   3443  0.0  0.0   3244   596 pts/0    S+   00:04   0:00 ./a.out
alireza   3444  0.0  0.0   3248   108 pts/0    S+   00:04   0:00 ./a.out
alireza   3457  0.0  0.0   5208  1188 pts/1    R+   00:05   0:00 ps -ux
alireza@alireza-Inspiron-N5010:~$
```



که PID ها, با PIDهایی که در اجرای برنامه‌یمان دیده بودیم برابر است.