السلام عليكم ورحمة الله وبركاته

اخباركم ايه يارب تكونوا بخير ..

programming-fr34ks.net ; linux arabic programming tutorial
author : St0rM-MaN
license : Rippers License , Gnu/Gpl ;
based on : advanced linux programming

OMG!!! Linux Programming !!!

Cool Ain't :D

Visit us (Http://Programming-Fr34ks.Net) ;

friends :
http://securitygurus.net
http://arab4services.com
http://linuxac.org
http://linux-fr34k.com
http://sechost-it.com
/*security rockers */

مقدمه اختياريه :

منذ مده كبيره وانا متردد , كل فتره تمر علي اقوم بفتح صفحه واسجل بها ملاحظات عامه ومتفرقه لتكون كمرجع لي , في مره من المرات حاولت ان انسقها لتصبح مقالمه كامله وصدقا لم استطع , ليس لاني فشلت بل لشدة الكسل , ايا كان ها قد قررت ان اكتب ولو مقال بسيط قد يساعدني في التذكر ان نسيت اولا وتكون لك كبدايه في البرمجه تحت نظام ال linux
لن اتطرق في الحديث عن قوة لينكس وعيوب ويندوز او العكس , ليس هناك اي داعي لذلك ولا اريد ان يصبح المقال لمناقشة من الافضل (بالعربي مش طالبه وجع دماغ) .

مقدمه اجباريه :
سيتم صدور عدة مقالات علي حسب تفرغي وعلي حسب وقتي (وحالتي النفسيه قبل اي شئ)
فليكن مايكون ها انت الان تقرا هذه السطور وانت تعلم اني قد كتبتها وانتهي الامر .
الموضوع بالتاكيد تحت الرخصه السفاحياه check out linux-fr34k.net وتحت الترتيب السفاحي اذا لاحظتم .

الموضوع حيتم عرضه علي هيئة سؤال وجواب وبعض القطع للمناقشه .
مقدم من : Programming-Fr34ks.net

اهدائات : كل اصدقائي (, strikerX , sofy , hacko , ray , mysql , dj ,gray ,abd elaziz acid) وكل من لم اذكر اسمائه منكم (تعرفون انفسكم فلا حاجه لذكركم)
لا تقرا السطور الاتيه اذا كنت من ذو المشاعر الحساساه ;

قلة ادب : ولاد الكلب الي عارفين نفسهم من اولهم لاخرهم الي بكرهمم اشد الكره بالعربي الصريح بقلوهم انتوا ارهابيين ولا ليكوا صله بالاسلام ولا بالامه الاسلاميه وبلاش شغل العيال الصغيره الي يعمل حاجه ويدراي ورا حاجه تانيه , كفايه نفاااق شوهتم صورة العرب والاسلام ينعل *** .***

الجزء الاول سوف يتم اصداره قريبا بس ال paper مش لاقيها الي كتبتها وحالتي مش تسمح اكتب واحده تانيه حاليا

برمجة اللينكس الجزء الثاني :

الفصل الاول : العمليات

س 1 : ماهي العمليات اصلا ؟
ج 1 : عباره عن التفيذ الخاص بالبرنامج الذي يعمل حاليا
اي برنامج في الدنيا يتكون من عدة statements و flows و و و يتم ترجمة هذا البرنامج من اللغه النصيه الخاصه بالبشر الي لغة الاله 0 1 01 0 , العمليه هي ماينفذ تلك التعليمات هي طور التفيذ نفسه عند بدا تشغيل اي برنامج تنشئ عمليه جديده
التعريف الاخير : هي عباره عن نسخه طبق الاصل من ال execution code اي الشفره التفيذيه الخاصه بالبرنامج

س 2 : لماذا نتعامل مع العمليات واين توجد ؟

في اي برنامج عندا تشفله تتكون عمليه جديده , تتعامل معها لكي تعطيك تحكم اكبر في برنامج واحد تستطيع به فعل عدة اشياء في وقت واحد ولا حاجه لكتابة اكثر من برنامج وتشغيلهم في وقت واحد ستقع في دائره كريهه ,.

س 3 : هل من المسموح برنامج ادارة اكثر من عمليه ؟
بالتاكيد , اكبر مثال علي ذلك هو linux kernel يدير اكثر من عمليه في نفس الوقت

س 4 : بالتاكيد يسمح لعمليه ادارة اكثر من برنامج صحيح ؟
بالطبع لا تاكد من اجابة س 1 وسوف تعرف كيف يمكن لعمليه ان تتعامل مع 2 program execution instance

س 5 : كيف يمكن تمميز عمليه عن الاخر ؟
لكل عمليه في نظام ال Linux معرف خاص بها يدعي ب process id اختصارا PID ماهو الا رقم عددي مكون من رقم طوله 16 BIT اي تحت نظام 32 BIT تكون 2 Byte .
تعرضت لذلك بالتاكيد من قبل ان كان لك خبره في التعامل مع النظام صحيح .

++ كل عمليه وانا اعني كل عمليه لها أب parent ماعدا عمليه واحده لها الرقم 0 PID تدعي ب INIT
هناك نوعين من العمليات نوع يسمي forked process و نوع اخر يسمي exec process

س 6 : ما الفرق بين fork , exec ؟
الفرق بسيط جدا اذا قرأت ال man pages لكل منها , forked process تاخذ صوره تبق الاصل من عدة اشياء من البرنامج مثل , signals , memory location , file descriptors , code flow .
اما ال exec process في معاكسة تمام لل forked . عندما يتم انشاء عمليه exec بواسطة exec functions family تتعرف اليها لاحقا يحدث التالي , ينحسر دور العمليه الجديده (\execution flow) وياتي دور برنامج جديد يضع ال (execution flow) الخاص به في تلك العمليه . (لا تهلك نفسك في محاولة فهم ذلك الان)

س 7 : اذا كنت اريد معرفة رقم العمليه الخاصه ببرنامج معين ماذا افعل ؟
حلين

1. ***pid_t getpid(void);***
2. ***ps command***

الحل الاول اذا كنت انت المبرمج وتريد معرفة رقم برنامجكك تقوم ببساطه بتعريف متغير من نوع pid_t وتقوم باستدعاء الداله اليه مثال : pid_t x = getpid();(يجب ان تستدعي المكتبه unistd.h

الحل الثاني اذا كنت تريد مرعرفة رقم اي عمليه مداره في ناظام ال linux الخاص بك
مثال علي ذلك : برنامج لا يفعل اي شيئ سوي دوره نهائيه . بالتاكيد كما قلت من قبل عن بدا تشغيل اي برنامج تنشئ عمليه جديده

```
#include<stdio.h>


int main(void)
{
   for(;;)
   ;

   return 0;
}
```

بعد ان تقوم بترجمته قم بتشغيله وافتح terminal اخر ونفذ الاتي :
ps -a -o pid,ppid,command,user,start_time
لن اشرح الامر كل ماريدك منك فعله مراجعة ال man pages الخاصه به
ايا كان انظر في القائمه التي ظهرت ستجد البرنامج الخاص بك تحت اسمه واسمك ووقت التشغيل ورقم العمليه الخاصه به ورقم العمليه الخاصه بال parent له .

س 8 : مذا استفيد اذن من كل هذا الهراء , اريد قفل ذلك البرنامج !!؟
كيف تقوم بقتل تلك العمليه ؟

kill command ,
man kill

kill pid signal
ببساطه خد رقم العمليه الخاصه بالبرنامج وقم بتفيذ الاتي
kill PID
وما ال signal ؟ سسناقش ذلك فيما بعد

س 9 : هل يمكن استخدام الامر kill من داخل البرنامج ؟
بالتاكيد man 3 kill الداله kill

```
#include<signal.h>

int kill(pid_t pid , int signal);
```

اذا كان : pid مساويا للصفر تقوم بارسال ال signal الي كل العمليات المتفرعه من ال group process id او ال GPID الخاصه بمرسل تلك ال signal .
اذا كانت اقل من الصفر : اذا كنت root او لديك صلاحيات super user سوف يتم الارسال الي كل العمليات . اذا كنت user عادي يتم ارسال ال signal الي كل العمليات التي تحمل ال user id الخاصه بك
اي انها سوف ترسل الي كل العمليات الخاصه بك . (استخدام ال root user خطر اذا كانت لا تعرف ما تفعله)
القيمه : 0 او -1 الاولي في النجاح والثانيه في الفشل يمكنك استخدام perror

راجع كل مافات جيدا لانه سوف يفيدك الان :

س 11 : في برنامج خاص بي كيف انشئ اكثر من عمليه واديرهم لاجعل كل واحده تفعل شيئ لانجز مهامي في وقت واحد ؟

يتوقف كل هذا عليك . هل تلك المهام المطلوبه منك تحتاج الي ان تبرمج كود خاص بك ؟ ام ان تلك المهام تحتاجك انت تقوم بتنفيذ عدة اوامر في نفس الوقت ؟

هناك نوعين من انشاء العمليات : استدعاء الداله system او استخدام for او exec او fork and exec
لا تستسهل الاولي حيث انها ليست جيده جدا لانها تقوم بعمل block لل process ال parent الخاصه بها في هذه الحاله برنامجكك الي حين ان تنتهي من تنفيذ البرنامج المطلوب او السطر المطلوب تعتمد علي /bin/sh وهذا مجرد link لل bash في كثير من الانظمه (واخري لا) .

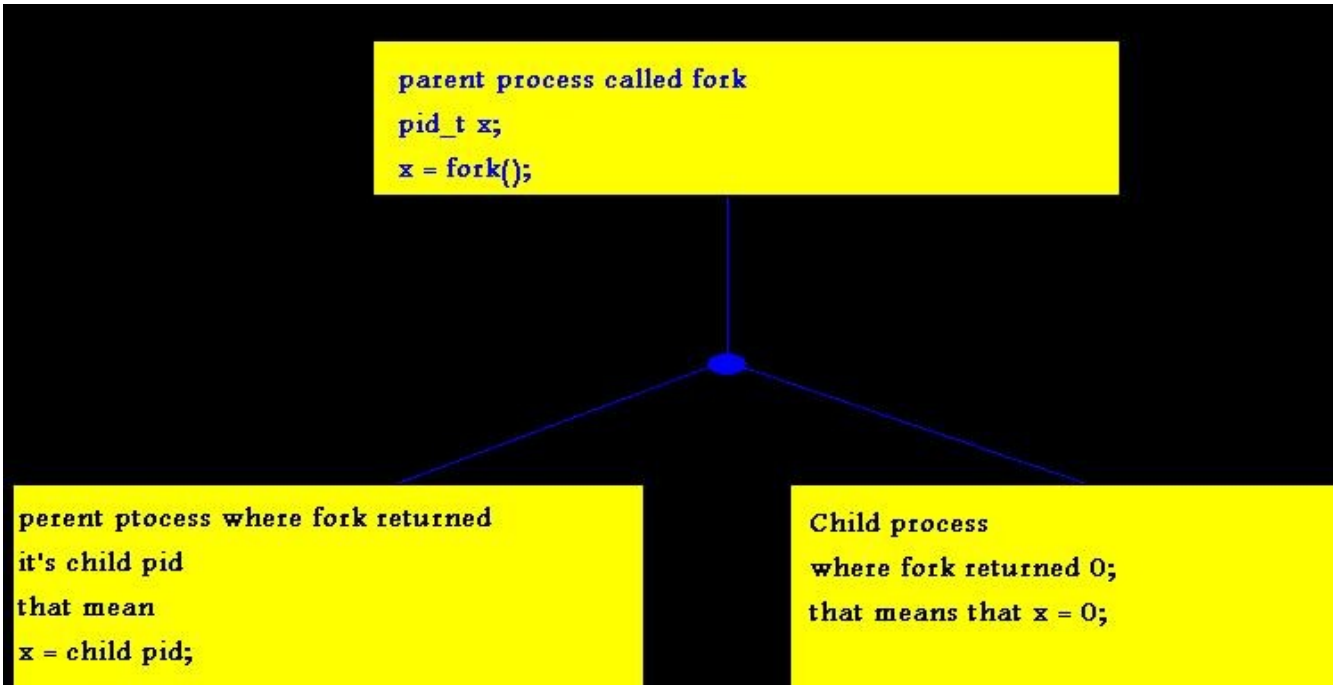
في الحاله الاولي سنتسخدم fork وليس exec لماذا ؟ راجع اجابة السؤال 6

حسنا لنخطط لذلك البرنامج
اريد انشاء عمليه جديده (لاحقا سوف تتعرف كيف تنشئ اكثر من عمليه (لاحظ انه هناك قيود لعدد العمليات قد تكون مفروضه عليك من قبل ال admin user الخاص بك))

قبل ان نقوم بانشاء هذا البرنامج تعرف اولا علي الاتي وادرسه :
واحد كيف تقوم fork بتفيذ مهامها

تقوم بعمل نسخه جديده من ال execution flow ألخاص بالبرنامج
تقوم بعمل pid جديد تحت مسمي cpid اي child pid
تقوم بتفيذ المهمه الجديده في ال child pid من بعد ما تم استدعائها اي تقوم بتفيذ الكود الذي يليها

تقوم تلك الداله بارجاع قيمتين !! كيف ليس هناك اي داله في لغة ال c تقوم براجاع قيمتين !!
انتظر لحظه تقوم في العمليه الاول برجاع قيمه وفي العمليه الثانيه بارجاع قيمه اخر تماما
في العمليه الاولي parent الي استدعتها في حالة اللنجاح تقوم بارجاع cpid اي child pid رقم عملية ال child الخاصه بها
في ال child pid تقوم براجاع 0 في حالة الفشل -1 ترجع وتوضع قيمه في errno يمكنك منها استدعء perror
جميل هل نرسم مخطط بسيط ؟



قد ذكرنا ان ال process المنشاءه من قبل fork تاخذ copy of it's parent execution code
يعني حتي كمان copy من ال x صح ؟
بكده fork تقدر تخليك تميز بين الاتنين .

س 12 : مامدة حياة تلك ال process ؟
لا نهائية الي انن يتم تنفيذ كافة البرنامج اي ان تلك العمليه لن تنتهي الا اذا اتمت عملها

س 12 : من الذي يموت الاول ال parent ام ال child ؟؟
اي منهما قد يموت الاول
لكن : اذا ماتت ال child فلن يكون هناك automatic cleaning يعني ال resources الي استخددمتها
مثال علي ذلك رقم ال pid سوف يظل محجوز لها بعد موتها ؟
او اذا ماتت ال parent ptocess سوف تتحول الي عمليه يتيمه ولكن لا تقلق عن ذلك سوف تتبناها
العمليه الاساسيه INIT كما ذكرنا من قبل .
لكن كيف تقوم بدفن تلك العمليه والتخلص من resources الخاصه بها ؟
انتظار موت تلك العمليه ؟ لا اعلم هل تمزح ؟ لا :D
هناك طريقتين للتنويه بان تلك العمليه ماتت
رقم واحد ; wait functions family سنستخدم منها حاليا wait فقط

```
#include<sys/types.h>
#include<sys/wait.h>

pid_t wait(int *ret_statues);
```

سوف تقوم تلك العمليه بايقاف تفيذ العمليه الرئيسيه الي تم الاستدعاء بها ثم وتنتظر موت العمليات الاخر , عندما تموت عمليه يتم ارجاع رقم تلك العمليه pid number ووضع قيمة الخروج الخاصه بتلك العمليه في ret_statues .

مثال كامل لانشاء عمليه وانتظار موتها
تلك العمليه سوف تنفذ function معينه كل وظيفتها طبعاة linux fr34ks.net -programming ...

ورقم عمليتها ورقم عملية ال parent الخاص بها ثم تقوم ب sleep لثانيه واحده ثم تنهي ال execution الخاص بها

العمليه الاصليه سوف تنتظر موت ال child الخاص بها ثم تقوم بطباعة كلمة والخروج ..

```
#include<stdio.h> /*statnder inpute/ouput functions*/
#include<unistd.h> /*fork , _exit */
#include<sys/types.h> /*pid_t */
#include<sys/wait.h> /*wait function */
#include<errno.h> /*errno global variable*/

void process(void)
{
    printf("Linux Programming Tutorial By Programming-Fr34ks[Dot]Net\n");
    printf("I Am The Child Process And This Is my Parent %d"
            " , And My Number %d\n",(int)getppid() , (int)getpid());
    sleep(2);
    _exit(0);
}

int main(void)
{
   int exit_statues;
   pid_t child_pid;

   /* create another process by calling fork
   *execution of this other process will start after fork
   * this process has a copy of it's parent address memory*/

   child_pid = fork (); /* --- here --- */

   if(child_pid == -1)
   {
            perror("fork()");
            _exit(1);
   }else if(child_pid = 0 ) /* this is the child process*/
   {
       process(); /*execute this function baby */
   }else /* not an error and not 0 this mean we are in parent process */
   {
       wait(&exit_statues); /* will block until a process dies */
   }
   /* here we are our child has died and we are in the execution flow of our main process*/

   printf("I Am The Main Process And i had A Child Called %d And It's Exit Statues Was %d"
           "And I Am %d\n", (int)child_pid , exit_statues ,(int) getpid());
   return 0;
}
```

الصراحه انا متوقع منك انك تكون فاهم الكلام ولا انا غلطان ؟

بالتاكيد طريقة الانتظار ديت ملعونه ! لانك لا تستفيد من ميزة ال multi-process !

س 13 : كيف تتغلب علي مشكلة الانتظار ؟
نحتاج الي طريقه وقتيه عندما تموت العمليه يتم ارسال اشاره لنا توا !
لحسن الحظ هذه موجود , عندما تموت عمليه يتم ارسال SIGCHLD كاشاره الي العمليه الاصليه
اذن نحتاج الي تعريف signal handler كي يقوم باستدعاء wait عندما تموت العمليه !
انتظر ؟ الم تقل ان wait تقوم بعمل block ؟ بلي ولكن اذا كانت العمليه ميته اصلا فليس هناك مايعدوا للانتظار !
راجع درس ال signals في programming-fr34ks.net

لنقم بكتابة كود جديد يستعدي نفس الداله الاخيره مع بعض التعديلات

```
#include<stdio.h> /*statnder inpute/ouput functions*/
#include<unistd.h> /*fork , _exit */
#include<sys/types.h> /*pid_t */
#include<sys/wait.h> /*wait function */
#include<signal.h> /*signal() , and signal identifiers*/
#include<errno.h> /*errno global variable*/


int exit_statues;

void sig(int sig_no)
{
    printf("Child Has Died\n");
    wait(&exit_statues);
}

void process(void)
{
    printf("Linux Programming Tutorial By Programming-Fr34ks[Dot]Net\n");
    printf("I Am The Child Process And This Is my Parent %d"
            " , And My Number %d\n",(int)getppid() , (int)getpid());
    sleep(10);
    _exit(0);
}

int main(void)
{
   signal(SIGCHLD , sig); /* announce that SIGCHLD will be handled by sig() function*/
   int i ; /*index variable */
   pid_t child_pid;

   /* create another process by calling fork
   *execution of this other process will start after fork
```

```
   * this process has a copy of it's parent address memory*/

   child_pid = fork (); /* --- here --- */

   if(child_pid == -1)
   {
            perror("fork()");
            _exit(1);
   }else if(child_pid = 0 ) /* this is the child process*/
   {
       process(); /*execute this function baby */
   }else /* not an error and not 0 this mean we are in parent process */
   {
       ; /* do nothing is this block */
   }
   /* we are in the  parent process */

      for(i=0 ; i < 15 ; i++)
     {
           printf("i am the parent process and i am doing something funny now\n");
           sleep(1); /*sleep for 1 second */
     }
     /* we are done from here and we know that our child has died
     *we specified for here a 10 seconde of execution remember sleep(10) in process
function()*/

   printf("I Am The Main Process And i had A Child Called %d And It's Exit Statues Was"
          "And I Am %d\n", (int)child_pid , exit_statues ,(int) getpid());
   return 0;
}
```

تعديلات بسيطه ولكن جوهريه ! قم بتنفيذ البرنامج سوف تطبع العباره i am the parent process and i am doing something funny now
10 مرات هل لاحظت اننا نقوم بعمل sleep كل دوره لمدة 1 ثانيه مما يعطيك 15 ثانيه من التفيذ لتلك العباره
اذن بعد 10 ثواني بعد ان تقوم process(); بالانتهاء سوف ترسل SIGCHLD الي ال parent المعرف بيها داله للتنفيذ عند تلك ال SIGNAL
س 14 : هل لل child process تاثير علي ال variables او ال memory الخاصه ب parent ?
لا بالتاكيد لانها تعتمد مبدا copy اي انها تحتوي فقط نسخه
مثال اخر عدلنا الكود الاخير حنستخدم x فيه مرتين في ال process وفي ال parent مرتين ونطبعها مره هنا ومره هنا شغل الكود وانت تشوف .

```
#include<stdio.h> /*statnder inpute/ouput functions*/
#include<unistd.h> /*fork , _exit */
#include<sys/types.h> /*pid_t */
#include<sys/wait.h> /*wait function */
#include<signal.h> /*signal() , and signal identifiers*/
#include<errno.h> /*errno global variable*/
```

```
int exit_statues;
int x = 12 ; /*some variable */
int i ;

void sig(int sig_no)
{
    printf("Child Has Died\n");
    wait(&exit_statues);
}

void process(void)
{
    printf("Linux Programming Tutorial By Programming-Fr34ks[Dot]Net\n");
    printf("I Am The Child Process And This Is my Parent %d"
            " , And My Number %d\n",(int)getppid() , (int)getpid());
            x = 2 ; /* x global variable in child process has the value 2 */
    for(i = 0 ; i < 10 ; i++)
    {

        printf("And X has %d in child process\n",x);
        sleep(1);
    }
    _exit(0);
}

int main(void)
{
   signal(SIGCHLD , sig); /* announce that SIGCHLD will be handled by sig() function*/
   pid_t child_pid;


   /* create another process by calling fork
   *execution of this other process will start after fork
   * this process has a copy of it's parent address memory*/

   child_pid = fork (); /* --- here --- */

   if(child_pid == -1)
   {
            perror("fork()");
            _exit(1);
   }else if(child_pid = 0 ) /* this is the child process*/
   {
       process(); /*execute this function baby */
   }else /* not an error and not 0 this mean we are in parent process */
```

```
  {
     x = 100 ; /* parent process will change value of x to 100 */
  }
  /* we are in the  parent process */

     for(i=0 ; i < 15 ; i++)
    {
          printf("i am the parent process and i am doing something funnny now"
                 "while x has %d in parent process\n",x);

          sleep(1); /*sleep for 1 second */
    }
    /* we are done from here and we know that our child has died
    *we specified for here a 10 seconde of execution remember sleep(10) in process
function()*/

  printf("I Am The Main Process And i had A Child Called %d And It's Exit Statues Was"
         "And I Am %d\n", (int)child_pid , exit_statues ,(int) getpid());
  return 0;
}
```

ها قد اتتمت ال fork section بالعربيه !

الم يحن الوقت ل exec function family ?

س 14 : مافائدة تلك الدوال ؟
كل ماتقوم به باستدعاء برنامج وتبديل ال execution code الخاص بالعمليه السابقه بال execution code للعمليه الجديده ! ليس لها اي علاقه بالعمليه السابقه !
كيف نستفيد منها ؟
نقوم بانشاء عمليه forked تحتوي علي نفس الشفره التنفيذيه القديمه
ثم نبدلها بشرفه تنفيذيه لبرنامج اخر وها اقد انتهينا :D
عدة functions موجوده لعدة jobs او methods دعنا نري
قاعدة عامه ال exec functions لاتقوم براجاع اي قيمه الا في حاله حدوث خطا !
مش حنحتاج منهم غير واحده بس execvp شوف ال man pages وانت تعرف الباقي
علي اي حال حعملهم list والسلام !

```
#include <unistd.h>

        extern char **environ;

        int execl( const char *path, const char *arg, ...);
        int execlp( const char *file, const char *arg, ...);
        int  execle( const char *path, const char *arg , ..., char
        * const envp[]);
        int execv( const char *path, char *const argv[]);
```

```
int execvp( const char *file, char *const argv[]);
```

اول واحده بتاخد مسار البرنامج بالتحديد + arguments + null terminated array of one argument
بالمناسبه زي ماي في اي برنامج argv[0] بتشير الي اسم البرنامج المفروض كمان arg المفروض
تشير الي اسم البرنامج المطلوب
execlp بتشير الي مكان معين + خاصية البحث في نفس ال directory الي تم استدعاء البرنامج منها
execle ولا ليها لزمه معانا لانها كل الي بتعمله بتستقبل environment variables يعني متغيرات بيئية
علي هيئية null terminated array of pointers لاحظ انها ثابته يعني بتتعرف مره واحده بس
مثلا PATH=~/Desktop
مثال

```
char *const envp[]  = {"PATH=~/Desktop",NULL};
```

execv اكتر واحده حنستخدمها بتستقبل المكان + array of pointers بتشير لل arguments
execvp الي فاتت بالظبت + خاصية البحث في نفس ال directory .

زي ماقلت ال functions مش بترجع اي قيمه الا في حالة الخطاء ترجع ب -1 + errno زي الي فاتت
يعني تقدر تستخدم perror
مثال كامل
برنامج بيقوم باستدعاء امر ال echo
و "programming-fr34ks.net" ك argument ليها .

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

int main(void)
{
   pid_t child_pid ;
   int exit_statues;

   child_pid = fork(); /*creat another process */
   const char *args[] = {"echo" , "programming-fr34ks[DoT]Net",NULL}; /* null
                                                    *terminated array of pointer*/
   if(child_pid == - 1 )
   {
            perror("fork()");
            _exit(1);
   }else if(child_pid = 0)
   {
       execvp("/bin/echo" , args);

       /*it will never return unless error occur then this steps will never be executed*/
       printf("Error Occur\n");
       perror("execvp()");
   }else /*parent*/
   {
       wait(&exit_statues);
   }
   printf("Child Terminatd %d\n",exit_statues);
```

```
    return 0;
}
```

انا تعبت خلاص ايديا وجعتني جدا وتقريبا ظهري اتحني 730 درجه :D وخلاااااااااااااااااااااااااص كده
اشوفكم في الجزء الثالث