Notas SO - *Execução de Programas*

"The exec family has many functions in C. These C functions are basically used to run a system command in a ***separate process that the main program*** *and print the output."*

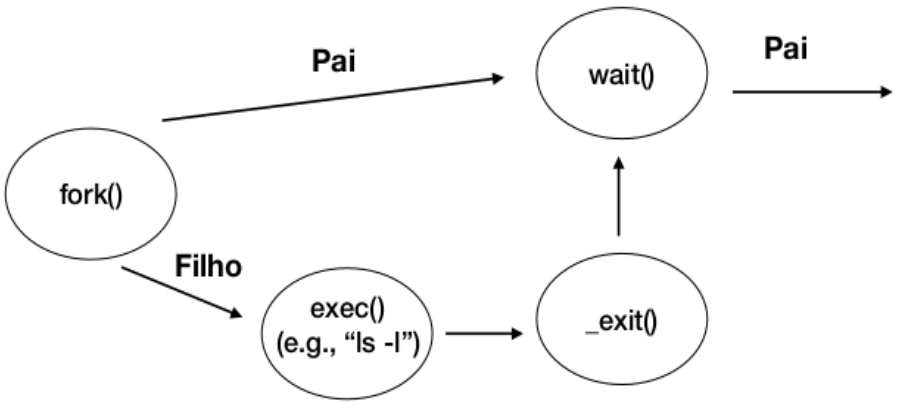
Definição

As primitivas *exec()* constituem uma família de funções: **execvp(...)**, **execle(...)**, **execve(...)**,**execl(...)**, **execlp(...)**,**execv(...)**.

Estas permitem o lançamento da execução de um programa externo ao processo. Não existe a criação efetiva de um novo processo, mas simplesmente uma substituição do programa de execução, sendo este especificado como argumento. Veremos as suas especificações de seguida. De realçar que, caso ocorra algum erro, **independentemente** da função em questão, então esta retorna **-1**. Caso contrário, não retorna **nada**.

Utilidade

Utilizando o sitema Fork + Exec, o processo pai cria um filho através do comando ***"Fork"*** e o filho executa um novo programa utilizando, para isso, o comando ***"Exec"***.



Especificação

execl()

```c
int execl(const char *path, const char *arg, ..., NULL);
```

- ☐ O **primeiro argumento** corresponde ao caminho completo para o executável;
- ☐ O **segundo argumento** corresponde, também, ao nome do executável, pois segue as normas *unix* em que o **arg[0]** é sempre o nome do executável;
- ☐ De seguida, ou seja a partir do **terceiro argumento**, inclusive, colocamos os argumentos que queremos passar;
- ☐ O **último argumento** terá **sempre** de ser ***NULL***;
- ☐ Ao executar, *imprime* o *output*.

```c
#include <unistd.h>

int main(void) {
  char *binaryPath = "/bin/ls";
  char *arg1 = "-lh";
  char *arg2 = "/home";

  execl(binaryPath, binaryPath, arg1, arg2, NULL);

  return 0;
}
```

execlp()

```
int execlp(const char *file, const char *arg,     , NULL );
```

- ☐ Semelhante à chamada anterior mas recorre aos caminhos registados na variável de ambiente *PATH*. Ou seja, comando ou o nome do ficheiro é suficiente para executar, sendo o *full path* dispensável;
  Exemplo: em vez de *"/bin/ls"*, o **primeiro** argumento **file** poderia ser apenas *"ls"*;
- ☐ O **último argumento** terá **sempre** de ser ***NULL***;
- ☐ Ao executar, *imprime* o *output*.

```
#include <unistd.h>

int main(void) {
  char *programName = "ls";
  char *arg1 = "-lh";
  char *arg2 = "/home";

  execlp(programName, programName, arg1, arg2, NULL);

  return 0;
}
```

execv()

```c
int execv(const char *path, char *const argv[]);
```

- O **primeiro argumento** corresponde ao caminho completo para o executável;
- A diferença desta função para as anteriores é que, em vez de passarmos diferentes num argumentos, podemos passar todos os parâmetros como um *array argv*;
- O **primeiro elemento** do *array argv* deverá ser o caminho para o executável. Caso contrário, então a função comporta-se da mesma forma que execl();
- A **última posição** do *array argv* terá **sempre** de ser ***NULL***;
- Ao executar, *imprime* o *output*.

```c
#include <unistd.h>

int main(void) {
  char *binaryPath = "/bin/ls";
  char *args[] = {binaryPath, "-lh", "/home", NULL};

  execv(binaryPath, args);

  return 0;
}
```

execvp()

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

- Funciona da mesma maneira que a função anterior, execv(), mas recorre aos caminhos registados na variável de ambiente *PATH*. Ou seja, comando ou o nome do ficheiro é suficiente para executar, sendo o *full path* dispensável;
  Exemplo: em vez de *"/bin/ls"*, o **primeiro** argumento **file** poderia ser apenas *"ls"*;
- O **primeiro elemento** do *array argv* deverá ser o comando ou o nome do ficheiro em questão;
- A **última posição** do *array argv* terá **sempre** de ser ***NULL***;
- Ao executar, *imprime* o *output*.

```c
#include <unistd.h>

int main(void) {
  char *programName = "ls";
  char *args[] = {programName, "-lh", "/home", NULL};

  execvp(programName, args);

  return 0;
}
```

execle()

```
int execle(const char *path, const char *arg, ..., NULL, char
    * const envp[] );
```

- ☐ Funciona da mesma maneira que a função execl(), mas com a diferença de que podemos fornecer as nossas próprias *environment variables* como um *array envp*;
- ☐ O **primeiro argumento** corresponde ao caminho completo para o executável;
- ☐ O **segundo argumento** corresponde, também, ao nome do executável, pois segue as normas *unix* em que o **arg[0]** é sempre o nome do executável;
- ☐ De seguida, ou seja a partir do **terceiro argumento**, inclusive, colocamos os argumentos que queremos passar. De realçar que é necessário colocar **NULL** no fim dos argumentos, ou seja, antes de passar o *array envp* ;
- ☐ A **última posição** do *array envp* terá **sempre** de ser ***NULL***. As restantes posições contém o par *key-value* como *String*;
- ☐ Ao executar, *imprime* o *output*.

```
#include <unistd.h>

int main(void) {
  char *binaryPath = "/bin/bash";
  char *arg1 = "-c";
  char *arg2 = "echo "Visit $HOSTNAME:$PORT from your browser
    .."";
  char *const env[] = {"HOSTNAME=www.linuxhint.com", "PORT
    =8080", NULL};

  execle(binaryPath, binaryPath,arg1, arg2, NULL, env);

  return 0;
}
```

execve()

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

- ☐ Funciona da mesma maneira que a função execle(), mas com a diferença de que podemos passar argumentos à função como um *array arg*, tal como fizemos na função execv();
- ☐ O **primeiro argumento** corresponde ao caminho completo para o executável;
- ☐ O **segundo argumento** corresponde, também, ao nome do executável, pois segue as normas *unix* em que o **arg[0]** é sempre o nome do executável;
- ☐ O **primeiro elemento** do *array argv* deverá ser o caminho para o executável;
- ☐ A **última posição** do *array argv* terá **sempre** de ser ***NULL***;
- ☐ A **última posição** do *array envp* terá **sempre** de ser ***NULL***. As restantes posições contém o par *key-value* como *String*;
- ☐ Ao executar, *imprime* o *output*.

```
#include <unistd.h>

int main(void) {
  char *binaryPath = "/bin/bash";
  char *const args[] = {binaryPath, "-c", "echo "Visit
   $HOSTNAME:$PORT
    from your browser."", NULL};
  char *const env[] = {"HOSTNAME=www.linuxhint.com", "PORT
   =8080", NULL};

  execve(binaryPath, args, env);

  return 0;
}
```

Síntese

```c
int execl(const char *path, const char *arg,     , NULL);

int execlp(const char *file, const char *arg,     , NULL );

int execv(const char *path, char *const argv[]);

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

int execle(const char *path, const char *arg,     , NULL, char
    * const envp[] );

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

-	*short path*	*full path*	*arg*	*argv[]*	*envp[]*
execl()	-	*Yes*	*Yes*	-	-
execlp()	*Yes*	-	*Yes*	-	-
execv()	-	*Yes*	-	*Yes*	-
execvp()	*Yes*	-	-	*Yes*	-
execle()	-	*Yes*	*Yes*	-	*Yes*
execve()	*Yes*	-	-	*Yes*	*Yes*