Exec functions

(simplified version)

Sistemas de Computadores

2020/2021

Luís Miguel Pinho / Luis Ferreira / Luís Nogueira / Nuno Pereira

exec functions

- Set of functions that allow for a process to execute a completely new program

- These functions replace the image of a process with another image, from a different program
  - The program being executed is replaced
  - But the process is still the same
    - Same PID
    - Same open files, …
    - But signals are reset to their default handlers!

exec functions

- It does not return to the previous program
  - It is replaced
- Only reason that the original program continues **if *exec* gives an error**
  - There is no return from a success execution of an *exec* function

execlp

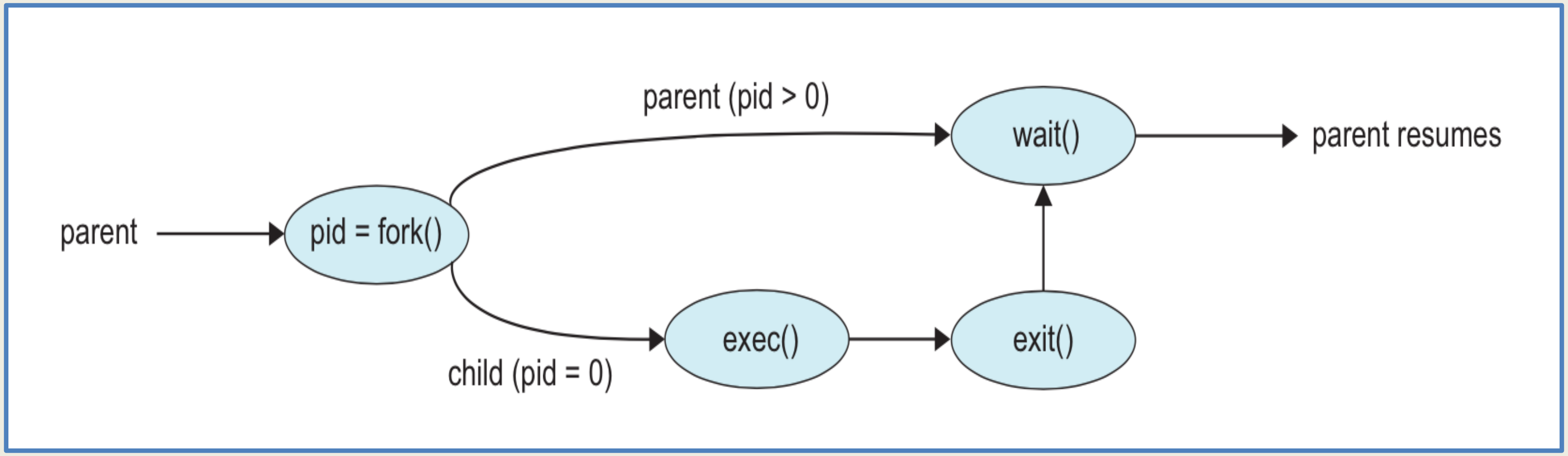
```
#include <unistd.h>

int main(){
   int ret;

   /* arg0 is the command name */
   ret = execlp ("ls", "ls", "-l", (char*)NULL);

   /* The program only reaches this point
       if execlp fails! */
   exit(ret);
}
```

Typical use of exec



Typical use of exec

```
pid = fork();

if(pid > 0){
  wait(&status);
  if(WIFEXITED(status))
    printf("Parent: child (exec) with
                exit value:%d\n", WEXITSTATUS(status));
   }
   else{
      execlp("prog","prog",(char*)NULL);
      exit(-1);    /* means exec failed */
   }
```

Exercise

- Considers the following program excerpt

```
fork();
fork();
for(i=0;i<5;i++)
   execlp("SCOMP","SCOMP",(char*)NULL);
```

- How many times SCOMP program is executed? Justify with the process tree

Exercise

- Considers the following program excerpt

```
for(i=0;i<3;i++){
   p = fork();
   x = fork();
   if(p>0 || x >0)
      execlp("SCOMP","SCOMP",(char*)NULL);
}
```

- How many times SCOMP program is executed? Justify with the process tree

OTHER EXEC FUNCTIONS

execl and *execv*

```
int execl(const char *path,   const
char *arg0, const char  *arg1, ...,
(char*)NULL);

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

execl and *execv*

- Replace with a program given by *path*
- *arg0* should have the name of the executable
- Only one difference between *execl* and *execv*:
  - *execl* receives the parameters with a list of arguments, ended with the NULL *string*
  - *execv* receives arguments in a vector of *strings;* last position must have the NULL *string*

execl

```
#include <unistd.h>

int main(){
   int ret;

   /* arg0 is the command name */
   ret = execl("/bin/ls", "ls", "-l”,
(char*)NULL);

   /* The program only reaches this point if
      execl fails! */
   exit(ret);
}
```

execv

```
#include <unistd.h>

int main(){
   int ret;

   char *cmd[] = {"ls", "-l", (char*)NULL};
   ret = execv("/bin/ls", cmd);

   /* The program only reaches this point
       if execv fails! */
   exit(ret);
}
```

execle and *execve*

```
int execle(const char *path,
const char *arg0, ...,(char*) NULL,
const *char envp[]);

int execve(const char *path,
   const char *args[] ,
   const  char *envp[]);
```

execle and *execve*

- Same behavior as previous functions

- Only adding the environment variables
  - Using *const *char envp[]*

execle

```
#include <unistd.h>

int main(){
   int ret;

   char *env[]={"HOME=/usr/home",
"LOGNAME=home", (char *)NULL};

   ret = execle ("/bin/ls", "ls", "-l", (char
*) NULL, env);

   exit(ret);
}
```

execlp and *execvp*

```
int execlp(const char *path,
   const char *arg0,...., (char*)NULL);

int execvp(const char *path,
   const char *args[] );
```

- If the full path is not specified, the executable is searched in the folders provided in the $PATH environment variable