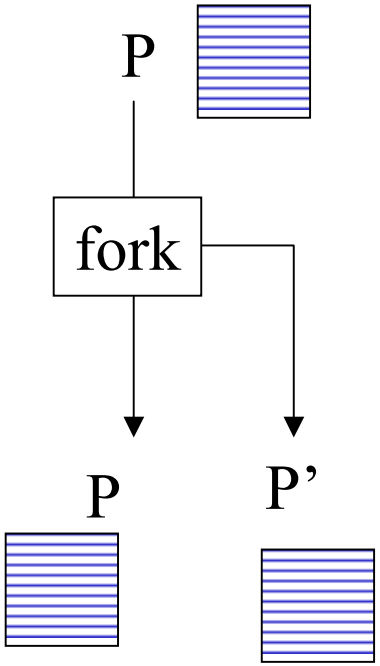
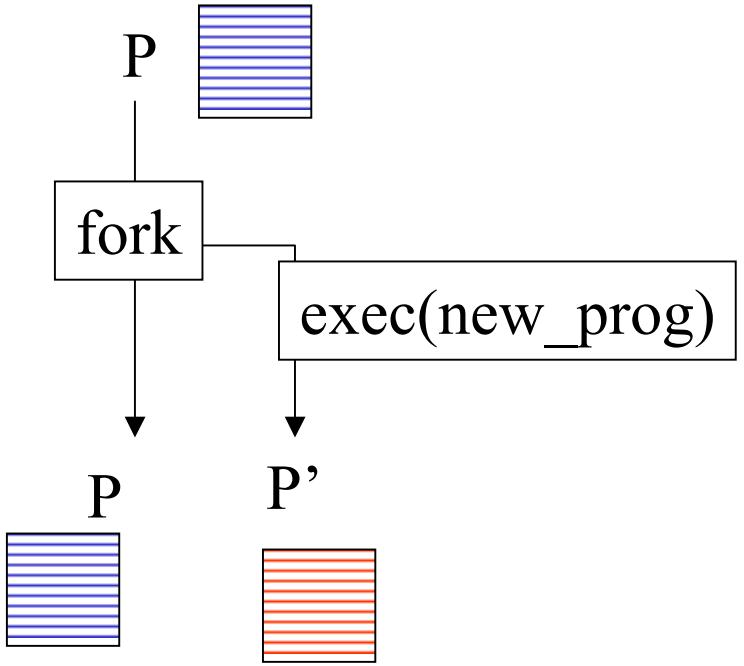
System Call EXEC

EXEC

L'effetto delle system call della famiglia exec consiste nel mandare in esecuzione: (1) un programma contenuto in un file eseguibile oppure un (2) interprete di programmi





Ho lasciato P' xché il PID del processo non cambia

Per noi sarà un eseguibile C

```
sono_il_padre = fork();
if (!sono_il_padre)  {
                        exec("new_prog");
                        /* il codice dopo exec non verrà
                           mai eseguito a meno che
                           exec fallisca  !!! */
                    }
<<codice padre>>
}
```

Non c'è ritorno da una exec

Non è una funzione terminata la quale si torna all'ambiente chiamante

```
main (int argc,  char **argv)
```

#argomenti ≥ 1
argv[0]=nome programma

argomenti
argv[1] … argv[argc-1]

```
#include <stdio.h>

main(int argc, char **argv)
{
 int i;

 printf("argc = %d\n", argc);
 for (i=0; i<argc; i++)
     printf("argv[%d] = %s\n", i, argv[i]);
}
```

```
prompt>prova 1 2 3
argc=4
argv[0]=prova
argv[1]=1
argv[2]=2
argv[3]=3
prompt>
```

EXEC

```
#include <unistd.h>

    int execl(const char *path, const char *arg0, ...,
                const char *argn, char * /*NULL*/);

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

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

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

    int execlp (const char *file, const char *arg0, ...,
                const char *argn, char * /*NULL*/);

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

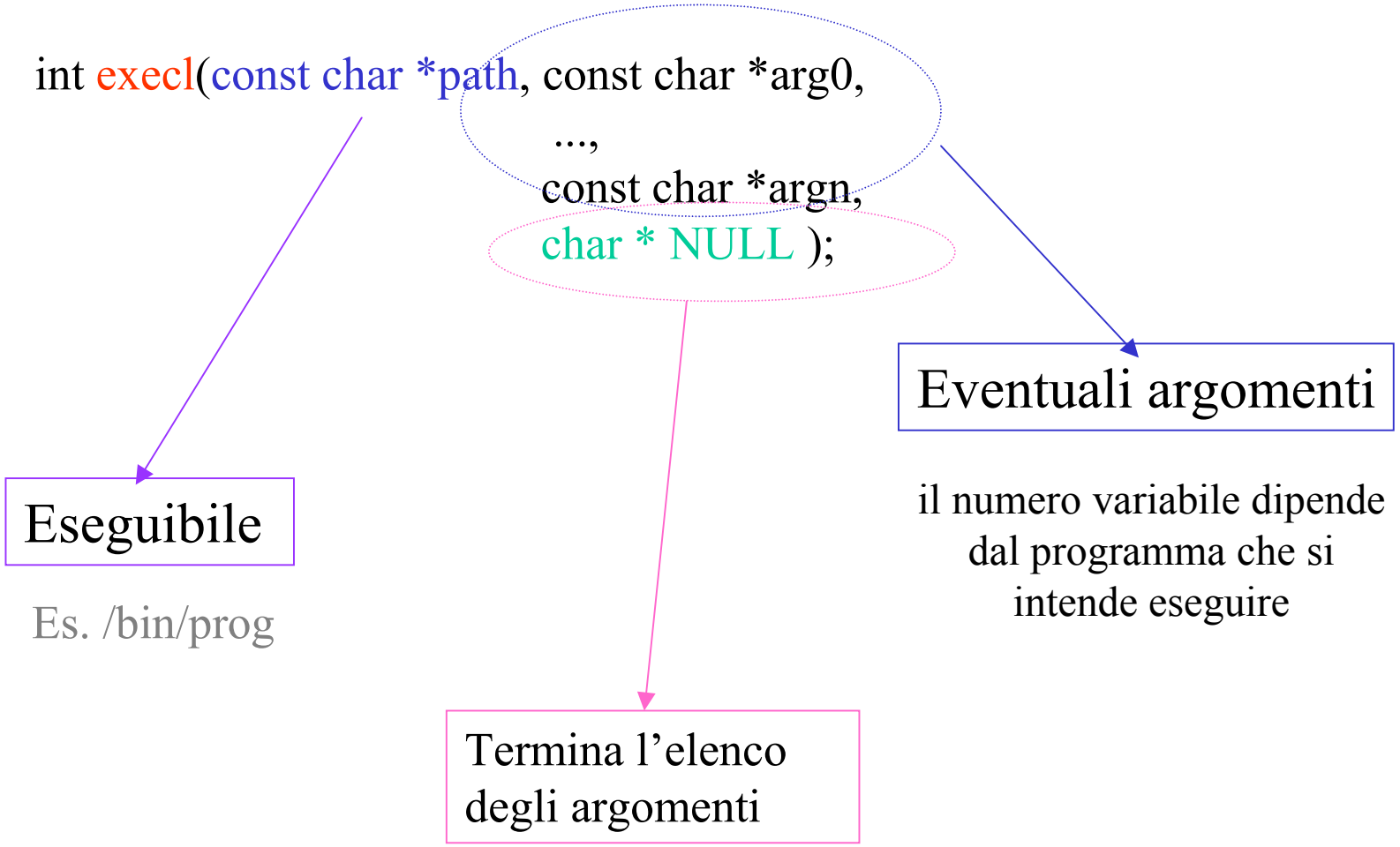
```
#include <sys/types.h>
#include <stdio.h>
main()
{
  pid_t  sono_il_padre;
  int s;

  if( (sono_il_padre=fork()) = = (pid_t) –1 )
       {fprintf(stderr, "fork fallita");  exit(1); }

  else if (sono_il_padre = = (pid_t) 0)
              {    /* processo figlio */
              execl("hello","hello",NULL);
              fprintf(stderr, "exec fallita");
              }
       else
              {    /* processo padre */
              wait(&s);
              }
}
```

Provare a eseguire due volte:

(1) Prima di avere compilato hello
(execl fallirà)

(2) Dopo aver compilato hello

```
main()
{
  printf("Hello \n");
}
```


int execl(const char *path, const char *arg0,
...,
const char *argn,
char * NULL);
Eseguibile
Es. /bin/prog
Termina l'elenco degli argomenti
Eventuali argomenti
il numero variabile dipende dal programma che si intende eseguire

Il processo mantiene:

nice value (see nice(2))
scheduler class and priority (see priocntl(2))
process ID
parent process ID
process group ID
supplementary group IDs
semadj values (see semop(2))
session ID (see exit(2) and signal(2))
trace flag (see ptrace(2) request 0)
time left until an alarm (see alarm(2))
current working directory
root directory
file mode creation mask (see umask(2))
resource limits (see getrlimit(2))
utime, stime, cutime, and cstime (see times(2))
file-locks (see fcntl(2) and lockf(3C))
controlling terminal
process signal mask (see sigprocmask(2))
pending signals (see sigpending(2))

All'utente che ha creato il processo vengono estesi i diritti d'accesso del proprietario del process file invocato tramite exec