Gómez Miranda Leopoldo

Tarea: Ejecución de exec

1.- execl
Parámetros:
l parámetros separados por comas
Forma:
int execl (char *path, char *arg_0, … char *arg_n, (char*)0);

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

main() {
pid_t pid;
int e;

        system("clear");
        pid=fork();

        if(pid==0){
                printf("execl \n");
                execl("/bin/ls", "ls", "-l", "-F",(char *)0);
                perror("fallo al intentar ejecutar ls");
        }
        else{
                wait(&e);
                exit(1);
        }
}
```

2.-execle
Parámetros:
l parámetros separados por comas
e variables de entorno pasadas como vector
Forma:
int execle (char *path, char *arg_0, … char *arg_n, (char*)0, char*envp[]);

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

main() {
pid_t pid;
int e;
char *env[] = { "TERM=xterm", (char *)0 };

        pid=fork();
        if(pid==0){
                execle("/usr/bin/clear","clear", (char *)0,env);
        }
        else{
                wait(&e);
                exit(1);
        }
}
/*
TERM, variable del shell
Contiene información sobre el tipo de terminal en uso, y lo utilizan
muchos programas para efectuar en forma correcta sus funciones de entrada
y salida.
*/
```

3.-execve
Parámetros:
v parámetros pasados como vector
e variables de entorno pasadas como vector
Forma:
int execve (char *path, char *argv[], char*envp[]);

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

main() {
pid_t pid;
int e;
char *arg[] = { "clear", (char *)0 };
char *env[] = { "TERM=xterm", (char *)0 };

        pid=fork();
        if(pid==0){
        execve("/usr/bin/clear", arg, env);
        }
        else{
                wait(&e);
                exit(1);
        }
}
/*
TERM, variable del shell
Contiene información sobre el tipo de terminal en uso, y lo utilizan
muchos programas para efectuar en forma correcta sus funciones de entrada
y salida.
*/
```

4.-execlp
Parámetros:
p busca la ruta del programa
l parámetros separados por comas
Forma:
int execlp (char *file char *arg_0, … char *arg_n, (char*)0);

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

```
main() {
pid_t pid;
int e;

        system("clear");
        pid=fork();

        if(pid==0){
                printf("execlp\n");
                execlp("ls", "ls", "-F", "-l", (char *)0);
                perror("fallo al intentar ejecutar ls");
        }
        else{
                wait(&e);
                exit(1);
        }
}
```

5.-execvp
Parámetros:
p busca la ruta del programa
v parámetros pasados como vector
Forma:
int execvp (char *file, char *argv[]);

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

main() {
pid_t pid;
int e;
char *arg[] = {"ls", "-F", "-l", (char *)0};

        system("clear");
        pid=fork();

        if(pid==0){
                printf("execvp \n");
```

```
                execvp("ls", arg);
                perror("execl fallo al intentar ejecutar ls");
        }
        else{
                wait(&e);
                exit(1);
        }
}
```

6.-execv
Parámetros:
v parámetros pasados como vector
Forma:
int execv (char *path, char *argv[]);

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

main() {
pid_t pid;
int e;
char *arg[] = {"ls", "-l", "-F",(char *)0};
        system("clear");
        pid=fork();

        if(pid==0){
                printf("execv\n");
                execv("/bin/ls",arg);
                perror("execl fallo al intentar ejecutar ls");
        }
        else{
                wait(&e);
                exit(1);
        }
}
```