Boletín 3: El comando `args`

Ampliación de Sistemas Operativos

Dpto. Ingeniería y Tecnología de Computadores (DITEC)

Universidad de Murcia

Curso 2017/2018

Índice

1. El comando `args`

El comando args

- (2.0p) **EJERCICIO 1**: Implementa el comando interno `args`:

```
simplesh> echo ESO > doc1.txt ; echo ISO > doc2.txt ; echo ASO > doc3.txt
simplesh> find . -name doc*.txt | sort | args wc
1 1 4 ./doc1.txt
1 1 4 ./doc2.txt
1 1 4 ./doc3.txt
```

- Para ello, únicamente se pueden usar llamadas al sistema POSIX o funciones de la librería estándar de C (C11)
- Llamadas `POSIX` y/o `glibc` a considerar:
  - `getopt()`
  - `read()`
  - `strchr()`
  - `fork()`
  - `exec*()`
  - `wait()` (`waitpid()`)
- Para implementar `args`, escribe la función `run_args()` e inserta llamadas a la misma en `simplesh.c` donde sea necesario

El comando args

- Si se proporciona el parámetro -h, args debe enviar a stdout:

```
simplesh> args -h
Uso: args [-d DELIMS] [-p NPROCS] [-h] [COMANDO [PARAMETROS] ]
            Opciones:
            -d DELIMS Caracteres delimitadores entre cadenas para COMANDO
            (defecto: " \t\r\n\v")
            -p NPROCS Número máximo de ejecuciones en paralelo de COMANDO
            (defecto: 1)
            -h help
```

- El comando por defecto es echo:

```
simplesh> echo ESO > doc1.txt ; echo ISO > doc2.txt ; echo ASO > doc3.txt
simplesh> find . -name doc*.txt | sort | args
./doc1.txt
./doc2.txt
./doc3.txt
```

- Por tanto, si se teclea sólo args:

```
simplesh> args
1234
1234
[CTRL+D]
simplesh>
```

El comando args

- El comando puede tener parámetros:

```
simplesh> echo ESO > doc1.txt ; echo ISO > doc2.txt ; echo ASO > doc3.txt
simplesh> find . -name doc*.txt | sort | args -- ls -l
-rw------- 1 alumno alumno 4 oct  1 18:00 ./doc1.txt
-rw------- 1 alumno alumno 4 oct  1 18:00 ./doc2.txt
-rw------- 1 alumno alumno 4 oct  1 18:00 ./doc3.txt
```

- El parámetro `-d` cambia los delimitadores por defecto (`WHITESPACE`):

```
simplesh> echo ESO > doc1.txt ; echo ISO > doc2.txt ; echo ASO > doc3.txt
simplesh> echo -n doc1.txt_doc2.txt%doc3.txt | args -d _% -- ls -l
-rw------- 1 alumno alumno 4 oct  1 18:00 ./doc1.txt
-rw------- 1 alumno alumno 4 oct  1 18:00 ./doc2.txt
-rw------- 1 alumno alumno 4 oct  1 18:00 ./doc3.txt
```

- Si el comando falla, `args` se detiene inmediatamente:

```
simplesh> echo ESO > doc1.txt ; echo ISO > doc2.txt ; echo ASO > doc3.txt
simplesh> echo -n doc1.txt_docX.txt_doc3.txt | args -d _ -- ls -l
-rw------- 1 alumno alumno 4 oct  1 18:00 ./doc1.txt
ls: no se puede acceder a 'docX.txt': No existe el archivo o el directorio
```

El comando args

- Implementación del comando `args`:
  1. Procesar los parámetros con `getopt()` (`man 3 getopt`)
     - **Nota**: Antes de volver a usar `getopt()`, se debe inicializar `optind` a `1`
  2. Leer una `CADENA` de la entrada estándar con `read()` hasta encontrar los delimitadores por defecto o los especificados por el usuario, o hasta que `read()` devuelva `0` notificando el fin del fichero (`EOF`)
     - La función `strchr()` comprueba si un carácter aparece en una cadena
  3. Crear un proceso hijo con `fork()` y ejecutar el comando por defecto o el especificado por el usuario en la línea de órdenes con `exec*()`
     - `fork( exec( echo CADENA ) )`
     - `fork( exec( COMANDO PARAMETROS CADENA ) )`
  4. Esperar a que finalice el proceso hijo creado con `wait()`
  5. Si el comando falla, es decir, si el proceso hijo devuelve un código de error, `args` se detiene inmediatamente

El comando `args`

- (0.5p) **Opcional**: El parámetro `-p NPROCS` especifica cuántos procesos pueden ejecutar `COMANDO PARAMETROS CADENA` en paralelo, es decir, `args` podría tener hasta `NPROCS` hijos simultáneamente
  - Por ejemplo: `(echo -n f1,f2,f3,f4,) | args -d , -p 2 - ls`
  - Nótese que `args` debe esperar a que terminen los hijos en orden de creación, detenerse en cuanto el primero de ellos falle y asegurarse de que no se generan procesos *zombie* en ningún caso