

Comunicación y tratado de ficheros secuencial

Jesús Rodríguez Heras

12 de mayo de 2019

Resumen

En este documento se desarrolla un tutorial de envío y recepción de ficheros mediante SSH entre los dispositivos del proyecto de manera secuencial y automática.

Índice

1. Introducción	3
2. Directorios	3
2.1. Directorio /backups	4
2.2. Directorio /recibir	4
2.3. Directorio /desencriptar	4
2.4. Directorio /trabajar	4
2.5. Directorio /enviar	5
3. Scripts	5
3.1. Recibiendo.sh	5
3.1.1. Código	5
3.1.2. Diagrama de flujo	7
3.2. Cristian.sh	8
3.2.1. Código	8
3.2.2. Diagrama de flujo	9
3.3. Enviando.sh	10
3.3.1. Código	10
3.3.2. Diagrama de flujo	12
3.4. Automatico.sh	12
3.4.1. Código	13
3.4.2. Diagrama de flujo	13
3.5. Borrar.sh	14
3.5.1. Código	14
3.5.2. Diagrama de flujo	14

1. Introducción

Para realizar una comunicación automática y secuencial entre los distintos dispositivos, cada tarjeta Zybo Zynq 7010 contará con la siguiente estructura de directorios:

```
~/ficheros/backups/  
|      |ViejoEnviar.txt  
|      |ViejoRecibir.txt  
|      |ViejoTrabajar.txt  
|desencriptar/..  
|enviar/..  
|recibir/..  
|trabajar/..  
|Automatico.sh  
|Borrar.sh  
|Cristian.sh  
|Enviando.sh  
|Recibiendo.sh
```

Este árbol de directorios estará en el archivo `ficheros.tar.gz` que se encontrará en el ordenador central y será distribuido a cada tarjeta mediante `ssh`¹ con el siguiente comando:

```
sshpass -p zyboX scp -o StrictHostKeyChecking=no ficheros.tar.gz  
zyboX@zyboX:
```

Luego, entramos en la tarjeta mediante `ssh` con el siguiente comando:

```
sshpass -p zyboX ssh -o StrictHostKeyChecking=no zyboX@zyboX
```

Poner alguna foto identificativa de lo que está pasando en todo el proceso

Una vez dentro del directorio `/home` de la tarjeta, aplicamos el siguiente comando para descomprimir el archivo `ficheros.tar.gz`:

```
tar -xzf ficheros.tar.gz
```

Y se nos creará el árbol de directorios anteriormente citado en el directorio `/home` de la tarjeta Zybo a la que hayamos accedido.

2. Directorios

En este apartado, describiremos los distintos directorios que podemos encontrar en las tarjetas Zybo una vez que se ha descomprimido el archivo `ficheros.tar.gz`.

Dicha descripción se hará siguiendo el orden que recorrerá el archivo recibido desde el ordenador central, salvo el directorio `/backups` que se describirá primero ya que no interviene de forma directa en el camino a recorrer por el archivo recibido.

¹Recordemos que, tanto para `ssh` como para `scp`, el elemento `zyboX` es el identificador de la tarjeta Zybo con la que estamos trabajando.

2.1. Directorio /backups

En este directorio se guardarán los estados de los siguientes directorios generados por el comando `stat`².

- **ViejoDesencriptar.txt:** Este fichero contendrá el estado del directorio `/desencriptar` una vez que el script `Desencriptando.sh` lo compruebe. Si no se producen cambios en dicho directorio, este fichero no se modificará.
- **ViejoEnviar.txt:** Este fichero contendrá el estado del directorio `/enviar` una vez que el script `Enviando.sh` lo compruebe. Si no se producen cambios en dicho directorio, este fichero no se modificará.
- **ViejoRecibir.txt:** Este fichero contendrá el estado del directorio `/recibir` una vez que el script `Recibiendo.sh` lo compruebe. Si no se producen cambios en dicho directorio, este fichero no se modificará.
- **ViejoTrabajar.txt:** Este fichero contendrá el estado del directorio `/trabajar` una vez que el script `Trabajando.sh` lo compruebe. Si no se producen cambios en dicho directorio, este fichero no se modificará.

2.2. Directorio /recibir

Este directorio contendrá los ficheros enviados por el ordenador central mediante `ssh`³.

Tendremos un script en segundo plano, `Recibiendo.sh`, que comprobará si hay algún cambio en el directorio. Si lo hay⁴, envía el archivo al directorio `/desencriptar`.

2.3. Directorio /desencriptar

Este directorio contendrá los ficheros enviados desde el directorio `/recibir` por el script `Recibiendo.sh` que comprobará el estado del directorio anterior.

Tendremos un script en segundo plano, `Cristian.sh` que comprobará si hay algún cambio en el directorio. Si lo hay, envía el archivo al directorio `/trabajar`.

2.4. Directorio /trabajar

Este directorio contendrá los ficheros enviados por el script `Cristian.sh` que comprobará el estado del directorio anterior.

Aquí, será también el script `Cristian.sh` el que tome partido, ya que para simular un tratamiento de datos, es este script el que añade al archivo una nueva línea diciendo que el archivo (inicialmente enviado por el ordenador central) ha sido tratado en la tarjeta en la que nos encontremos.

Concretamente, la línea que introduce en el archivo es la siguiente:

Archivo tratado en zyboX

Siendo `zyboX` el identificador de la tarjeta con la que estamos trabajando.

²Para más información leer el manual del comando `stat` en este [enlace](#).

³Para ver como enviar ficheros desde el ordenador central hasta este directorio, ver el documento “Envío y recepción de ficheros con `sshpass`”.

⁴Si hay un cambio en el directorio, supone que ha llegado un nuevo archivo por parte del ordenador.

2.5. Directorio /enviar

Este directorio contendrá los ficheros enviados por el script `Cristian.sh` que comprobará el estado del directorio anterior.

Aquí también tendremos un script en segundo plano, `Enviando.sh`, que comprobará si hay algún cambio en los ficheros y los enviará al siguiente dispositivo mediante `ssh`⁵.

3. Scripts

En este apartado, describiremos los distintos scripts que podemos encontrar en las tarjetas Zybo con su descripción y código correspondiente.

Dicha descripción se hará siguiendo el orden que recorrerá el archivo recibido desde el ordenador central.

Para comprobar el estado de todos los directorios, usaremos el comando `stat` (anteriormente mencionado en este documento) que nos dará la información de los ficheros que existen en un directorio.

Debemos tener en cuenta que, para emular el trabajo de nuestro compañero Cristian, hemos hecho que el script `Cristian.sh` sea el que “desencripta” el fichero, añade información (tal como está en 2.4) y “encripta” el fichero de nuevo, justo antes de enviarlo al directorio `/enviar`.

3.1. Recibiendo.sh

Este script es el encargado de comprobar periódicamente el estado del directorio `/recibir` y, si llega un archivo nuevo, enviarlo al directorio `/desencriptar`.

3.1.1. Código

```
1  #!/bin/bash
2
3  #Recibiendo.sh
4
5  DIR_TO_CHECK=$PWD/recibir #Directorio que queremos inspeccionar
6
7  DIR_TO_SEND=$PWD/desencriptar #Directorio donde enviar
8
9  OLD_STAT_FILE=$PWD/backups/ViejoRecibir.txt #Estado del directorio
10
11 if [ -e $OLD_STAT_FILE ]
12 then
13     OLD_STAT=`cat $OLD_STAT_FILE`
14 else
15     OLD_STAT="nothing"
16 fi
17
18 NEW_STAT=`stat -t $DIR_TO_CHECK`
```

⁵Para ver como funciona el envío de ficheros mediante `ssh`, ver el documento “Envío y recepción de ficheros con `sshpass`”.

```

19
20 if [ "$OLD_STAT" != "$NEW_STAT" ]
21 then
22
23     cd $DIR_TO_CHECK
24
25     n=$(ls | wc -l)
26     cero=0
27
28     if [[ n -ne cero ]]
29     then
30         fichero=$(ls -t | head -1) #Obtenemos el fichero más reciente
31         cp $fichero $DIR_TO_SEND
32     fi
33     echo $NEW_STAT > $OLD_STAT_FILE #Actualiza el estado del
        directorío
34 fi

```

Código de Recibiendo.sh.

3.1.2. Diagrama de flujo

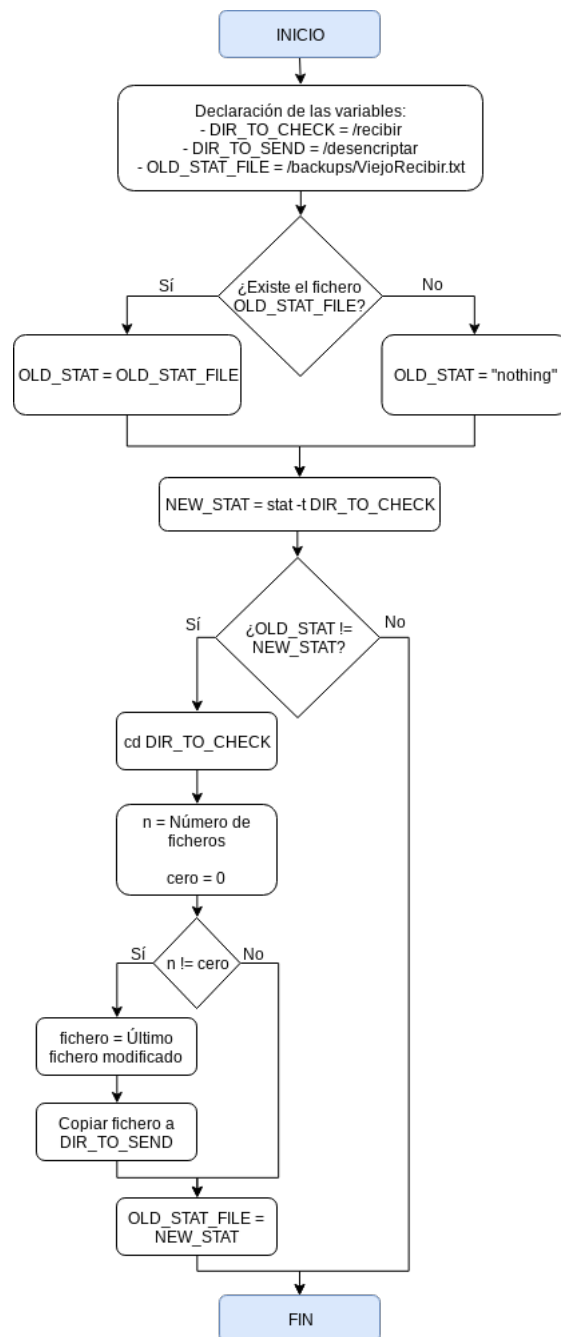


Figura 1: Diagrama de flujo de `Recibiendo.sh`.

3.2. Cristian.sh

Este script es el encargado de emular el trabajo de nuestro compañero Cristian. Se encarga de comprobar periódicamente el estado del directorio /desencriptar, mueve el archivo allí situado al directorio /trabajar (simulando el desencriptado) y allí le añade un texto tal como el siguiente:

Archivo tratado en zyboX

Siendo zyboX el identificador de la tarjeta con la que estamos trabajando.

Por último, envía el fichero al directorio /enviar.

3.2.1. Código

```
1  #!/bin/bash
2
3  #Cristian.sh
4
5  DIR_TO_CHECK=$PWD/desencriptar #Directorio que queremos inspeccionar
6
7  DIR_TEMP=$PWD/trabajar #Directorio temporal para trabajar
8
9  DIR_TO_SEND=$PWD/enviar #Directorio donde enviar
10
11 OLD_STAT_FILE=$PWD/backups/ViejoTrabajar.txt #Estado del directorio
12
13 if [ -e $OLD_STAT_FILE ]
14 then
15     OLD_STAT=`cat $OLD_STAT_FILE`
16 else
17     OLD_STAT="nothing"
18 fi
19
20 NEW_STAT=`stat -t $DIR_TO_CHECK`
21
22 if [ "$OLD_STAT" != "$NEW_STAT" ]
23 then
24
25     cd $DIR_TO_CHECK
26
27     n=$(ls | wc -l)
28     cero=0
29
30     if [[ n -ne cero ]]
31     then
32         fichero=$(ls -t | head -1) #Obtenemos el fichero más reciente
33         cp $fichero $DIR_TEMP
34         cd $DIR_TEMP
35         #Trabajamos con el fichero
36         echo 'Archivo tratado en '$USER >> $fichero
```



```

37         cp $fichero $DIR_TO_SEND
38     fi
39     echo $NEW_STAT > $OLD_STAT_FILE
40 fi

```

Código de Cristian.sh.

3.2.2. Diagrama de flujo

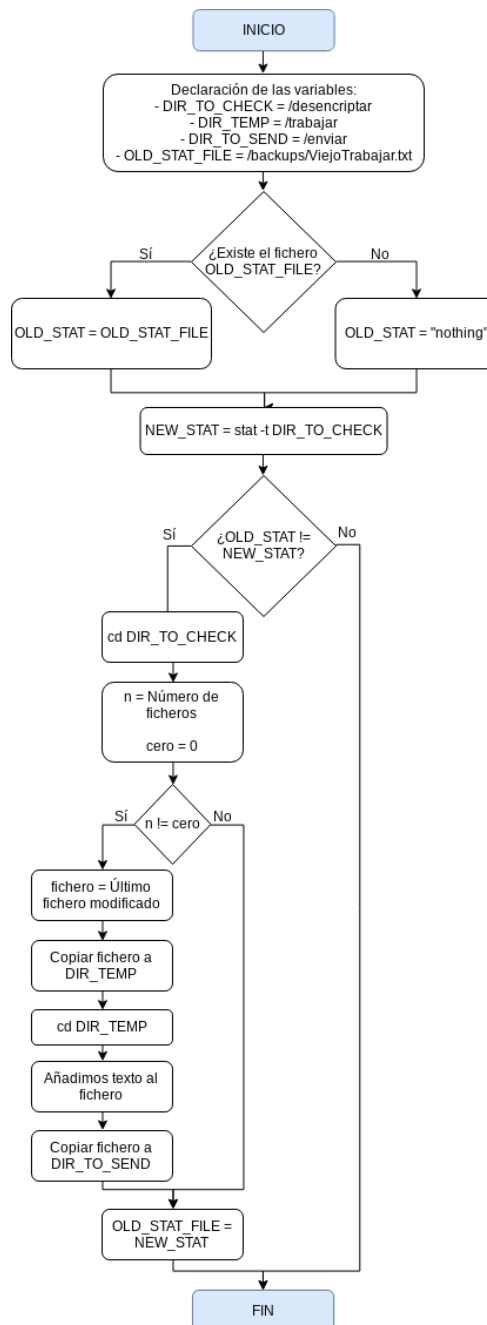


Figura 2: Diagrama de flujo de Cristian.sh.

3.3. Enviando.sh

Este script es el encargado de comprobar periódicamente el estado del directorio `/enviar` y, cuando detecta un cambio, envía el archivo a la siguiente tarjeta, o, si ésta se encuentra desconectada, al ordenador central.

A la hora de comprobar si la siguiente tarjeta está conectada o no, se hace enviando un comando ping a la siguiente tarjeta.

Para que podamos usar el comando ping desde este script, debemos darle permisos de ejecución en modo usuario de la siguiente forma:

1. Entramos como super-usuario con el comando `su` y contraseña `zyboX` (siendo X el identificador de la tarjeta con la que estamos trabajando).
2. A continuación, introducimos el siguiente comando:

```
chmod u+s /bin/ping
```

Y con eso, quedaría activado el comando `ping` para poder usarlo desde este script.

3.3.1. Código

```
1  #!/bin/bash
2
3  #Enviando.sh
4
5  DIR_TO_CHECK=$PWD/enviar #Directorio que queremos inspeccionar
6
7  OLD_STAT_FILE=$PWD/backups/ViejoEnviar.txt #Estado del directorio
8
9  if [ -e $OLD_STAT_FILE ]
10 then
11     OLD_STAT=`cat $OLD_STAT_FILE`
12 else
13     OLD_STAT="nothing"
14 fi
15
16 NEW_STAT=`stat -t $DIR_TO_CHECK`
17
18 if [ "$OLD_STAT" != "$NEW_STAT" ]
19 then
20
21     cd $DIR_TO_CHECK
22
23     n=$(ls | wc -l)
24     cero=0
25
26     if [[ n -ne cero ]]
27     then
28         fichero=$(ls -t | head -1) #Obtenemos el fichero más reciente
29
```

```

30     actual=$USER
31     nActual=$(echo ${actual##*o})
32     let nSiguiente=nActual+1
33     siguiente=zybo$nSiguiente
34
35     string=$(ping -c 3 $siguiente)
36
37     if [[ $string == *100%\ packet\ loss* ]]
38     then
39         sshpass -p zybomonitor scp -o StrictHostKeyChecking=no
40             $fichero zybo@monitor:/home/zybo/Documentos/Zybo
41     else
42         siguienteZybo=$siguiente
43         siguienteZybo+=@
44         siguienteZybo+=${siguiente}
45         siguienteZybo+=:/home/
46         siguienteZybo+=${siguiente}
47         siguienteZybo+=/ficheros/recibir
48
49         sshpass -p $siguiente scp -o StrictHostKeyChecking=no
50             $fichero $siguienteZybo
51     fi
52     echo $NEW_STAT > $OLD_STAT_FILE
53 fi

```

Código de Enviando.sh.

3.3.2. Diagrama de flujo

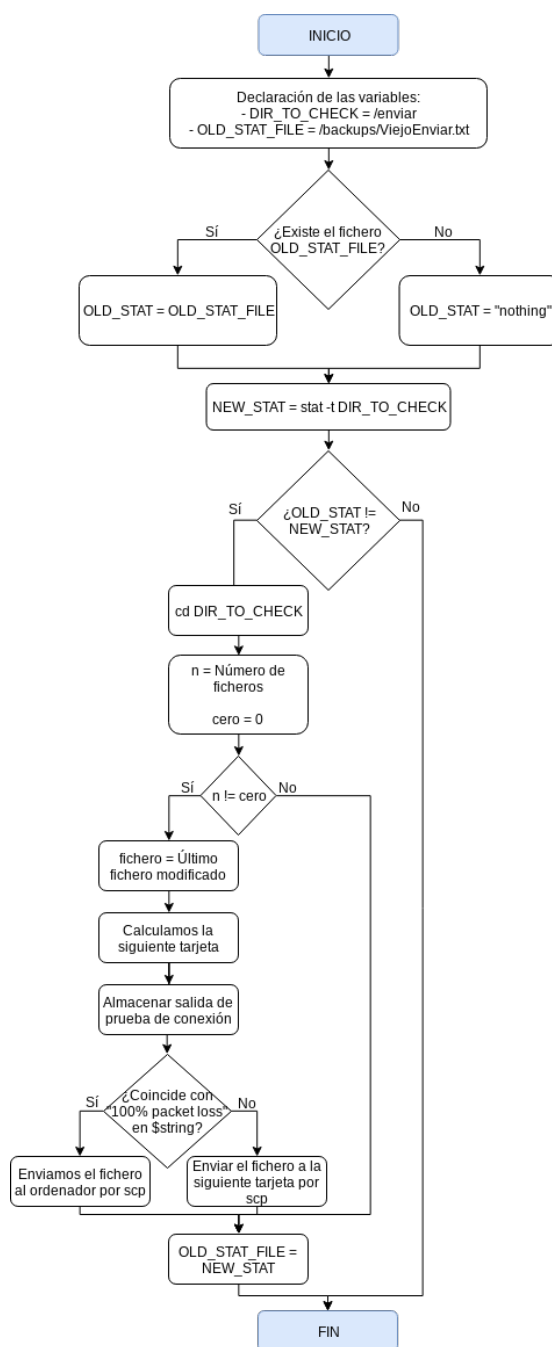


Figura 3: Diagrama de flujo de Enviando . sh.

3.4. Automatico.sh

Este script es el encargado de lanzar el resto de scripts en segundo plano cada segundo para que vayan comprobando los directorios correspondientes y se produzca la comunicación de forma automática.

Para ejecutarlo solo debemos usar el siguiente comando en el directorio `/ficheros` de las tarjetas Zybo:

```
./Automatico.sh &
```

De esta forma, lo mandamos a segundo plano, para poder seguir utilizando la consola de las tarjetas con normalidad.

3.4.1. Código

```
1 #!/bin/bash
2
3 #Automatico.sh
4
5 while :
6 do
7     ./Recibiendo.sh &
8     ./Cristian.sh &
9     ./Enviando.sh &
10    sleep 1
11 done
```

Código de `Automatico.sh`.

3.4.2. Diagrama de flujo

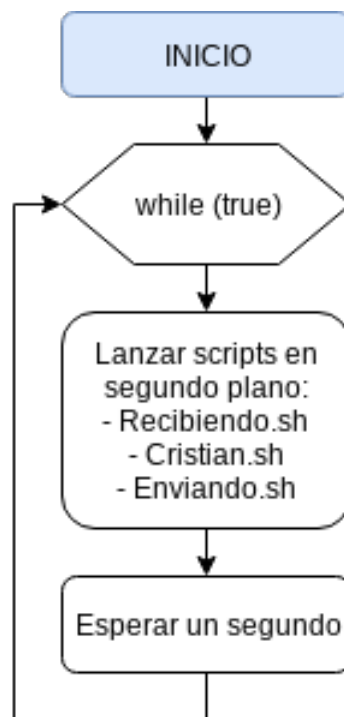


Figura 4: Diagrama de flujo de `Automatico.sh`.

3.5. Borrar.sh

Este script se encarga de vaciar los directorios recibir, desencriptar, trabajar y enviar de las tarjetas Zybo.

Para ejecutarlo solo debemos usar el siguiente comando en el directorio /ficheros de las tarjetas Zybo:

```
./Borrar.sh
```

3.5.1. Código

```
1 #!/bin/bash
2
3 #Borrar.sh
4
5 rm recibir/*
6 rm desencriptar/*
7 rm trabajar/*
8 rm enviar/*
```

Código de Borrar.sh.

3.5.2. Diagrama de flujo



Figura 5: Diagrama de flujo de Borrar.sh.