

# Comunicación y tratado de ficheros secuencial

Jesús Rodríguez Heras

15 de mayo de 2019

## Resumen

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

Para conseguir dicha finalidad, las tarjetas iniciarán automáticamente un proceso que comprobará si les envían un fichero nuevo y, en caso de que así sea, realizarán alguna modificación en él y lo enviarán a la siguiente tarjeta, o, en caso de que no esté conectada, al ordenador central.

- Particularmente, la tarea a realizar aquí consta de la recepción de un fichero cifrado, su descifrado, su modificación, su cifrado y, por último su envío hacia el siguiente dispositivo.
- Se describirá una estructura de directorios que serán recorridos por el fichero inicial en sus distintos estados.

Basándonos en dichos estados, nombraremos dichos directorios tal como se describen a continuación.

- El fichero será recibido en el directorio `~/ficheros/recibir` e irá pasando por los directorios `~/ficheros/desencriptar`, `~/ficheros/trabajar` y `~/ficheros/enviar` antes de ser enviado al siguiente dispositivo.

# Índice

<b>1. Introducción</b>	<b>3</b>
<b>2. Directorios</b>	<b>4</b>
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
<b>3. Scripts</b>	<b>5</b>
3.1. Lanzador.sh . . . . .	5
3.1.1. Diagrama de flujo . . . . .	6
3.1.2. Código . . . . .	6
3.2. Automatico.sh . . . . .	6
3.2.1. Diagrama de flujo . . . . .	7
3.2.2. Código . . . . .	7
3.3. Recibiendo.sh . . . . .	7
3.3.1. Diagrama de flujo . . . . .	8
3.3.2. Código . . . . .	8
3.4. Cristian.sh . . . . .	9
3.4.1. Diagrama de flujo . . . . .	10
3.4.2. Código . . . . .	11
3.5. Enviando.sh . . . . .	12
3.5.1. Diagrama de flujo . . . . .	13
3.5.2. Código . . . . .	14
3.6. Borrar.sh . . . . .	15
3.6.1. Diagrama de flujo . . . . .	15
3.6.2. Código . . . . .	16

# 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`<sup>1</sup> 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
```

Para ver si tenemos el archivo `ficheros.tar.gz` usamos el comando:

```
ls
```

A continuación, aplicamos el siguiente comando para descomprimir el archivo `ficheros.tar.gz`:

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

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

Para que la automatización del proceso se lleve a cabo correctamente, también tendremos que modificar el fichero `/etc/hosts` de la tarjeta. Para ello, enviamos el fichero con el siguiente comando:

```
sshpass -p zyboX scp -o StrictHostKeyChecking=no hosts
zyboX@zyboX:
```

Y, luego, lo copiamos como super-usuario<sup>2</sup> en el directorio `/etc` con el siguiente comando:

```
cp hosts /etc
```

Una vez hecho esto, el proceso de automatización estaría listo para ser lanzado.

---

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

<sup>2</sup>Comando: `su`, y contraseña `root`.

## 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.

### 2.1. Directorio `/backups`

En este directorio se guardarán los últimos estados de los directorios nombrados a continuación, generados por el comando `stat`<sup>3</sup>.

- **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 otros dispositivos mediante `ssh`<sup>4</sup>.

### 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.

### 2.4. Directorio `/trabajar`

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

Será donde se realicen las acciones específicas que se contemplan en este tutorial.

---

<sup>3</sup>Para más información leer el manual del comando `stat` en este [enlace](#).

<sup>4</sup>Para ver como enviar ficheros desde un dispositivo a otro hasta este directorio, ver el documento “Envío y recepción de ficheros con `sshpass`”.

## 2.5. Directorio /enviar

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

Una vez aquí, dichos ficheros estarán listos para pasar al siguiente dispositivo.

## 3. Scripts

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

Dicha descripción se hará siguiendo el orden que recorrerá el archivo recibido desde el dispositivo anterior hasta ser enviado al siguiente dispositivo.

Para comprobar el estado de todos los directorios, usaremos el comando `stat` para comprobar el estado de los directorios.

En el trabajo aquí mencionado se emula el desenscriptado de un fichero, adición de información, cifrado, y envío del mismo a otro dispositivo<sup>5</sup>.

### 3.1. Lanzador.sh

Este script se encarga de lanzar el script `Automatico.sh` mediante la herramienta `cron`<sup>6</sup> al inicio del sistema operativo Xillinux.

Para usarlo, debemos usar el siguiente comando:

```
crontab -e
```

Y, luego, añadir la regla que queramos que se ejecute al final del fichero. En nuestro caso es la siguiente:

```
@reboot (cd ~/ficheros; ./Lanzador.sh)
```

Esto hará que la herramienta `cron` inicie este script al iniciar el sistema operativo Xillinux de las tarjetas Zybo.

---

<sup>5</sup>Para cambiar dicho comportamiento, solo tendremos que modificar los scripts que se encargan de automatizar el proceso.

<sup>6</sup>Para más información, ver el manual de `crontab` en este [enlace](#).

### 3.1.1. Diagrama de flujo

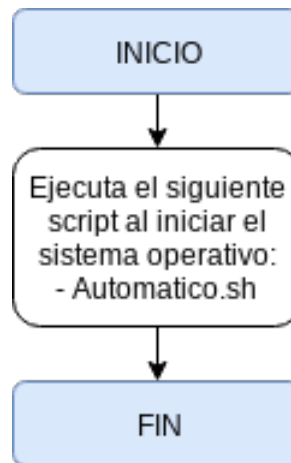


Figura 1: Diagrama de flujo de `Lanzador.sh`.

### 3.1.2. Código

```
1 #!/bin/bash
2
3 #Lanzador.sh
4
5 ./Automatico.sh
```

Código de `Lanzador.sh`.

## 3.2. Automatico.sh

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

### 3.2.1. Diagrama de flujo

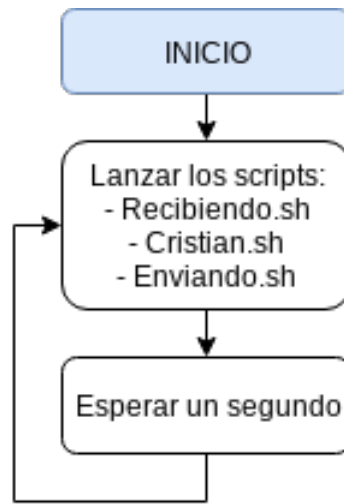


Figura 2: Diagrama de flujo de Automatico.sh.

### 3.2.2. 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.

### 3.3. Recibiendo.sh

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

### 3.3.1. Diagrama de flujo

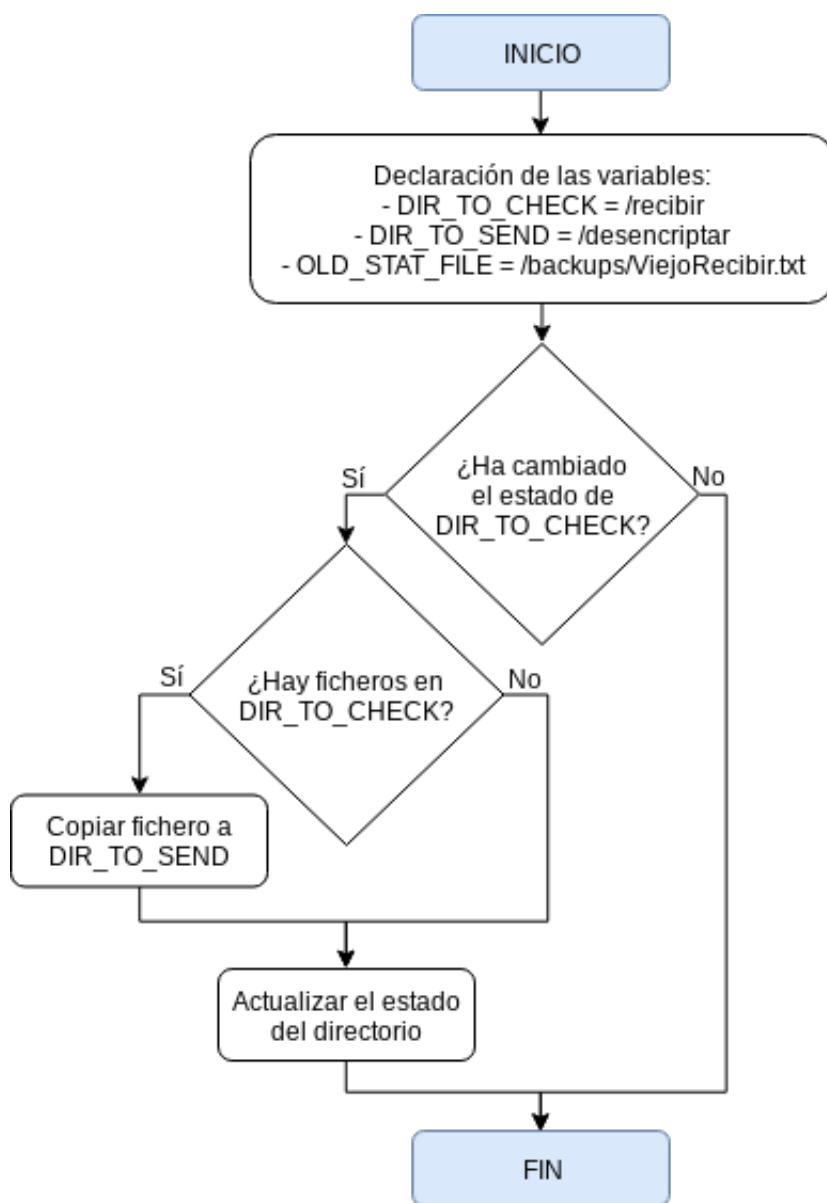


Figura 3: Diagrama de flujo de `Recibiendo.sh`.

### 3.3.2. 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 $DIR_TO_CHECK`
19
20 if [ "$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 directorio
34 fi

```

Código de Recibiendo.sh.

### 3.4. Cristian.sh

Este script es el encargado de emular el trabajo de nuestro compañero Cristian y realiza las siguientes tareas:

- Gracias al crontab establecido, se encarga de comprobar periódicamente el estado del directorio ~/ficheros/desenciptar.
- Mueve el archivo allí situado al directorio ~/ficheros/trabajar (simulando el desencryptado del mismo).
- Una vez allí, añade un texto 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 ~/ficheros/enviar.

### 3.4.1. Diagrama de flujo

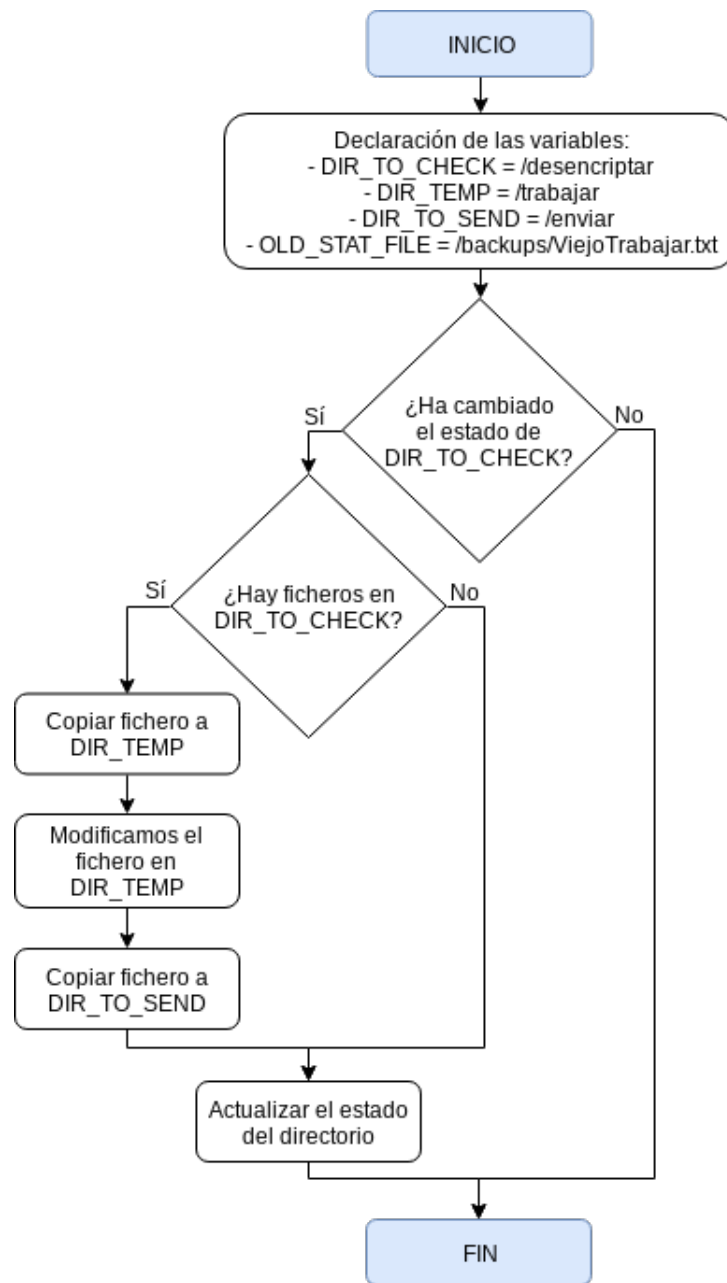


Figura 4: Diagrama de flujo de `Cristian.sh`.

### 3.4.2. 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 $DIR_TEMP
35         tarjeta=$(cat /etc/passwd | cut -d : -f1 | grep zybo)
36         echo 'Archivo tratado en '$tarjeta >> $fichero
37         cp $fichero $DIR_TO_SEND
38     fi
39     echo $NEW_STAT > $OLD_STAT_FILE #Actualiza el estado del directorio
40 fi
```

Código de Cristian.sh.

### 3.5. Enviando.sh

Este script es el encargado de comprobar periódicamente el estado del directorio `~/ficheros/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, por lo que se nos presentarán dos posibles casos:

- **Éxito:** La tarjeta está conectada y será allí donde se envíe el fichero.
- **Fracaso:** La tarjeta no está conectada y el fichero será enviado al ordenador central.

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.

### 3.5.1. Diagrama de flujo

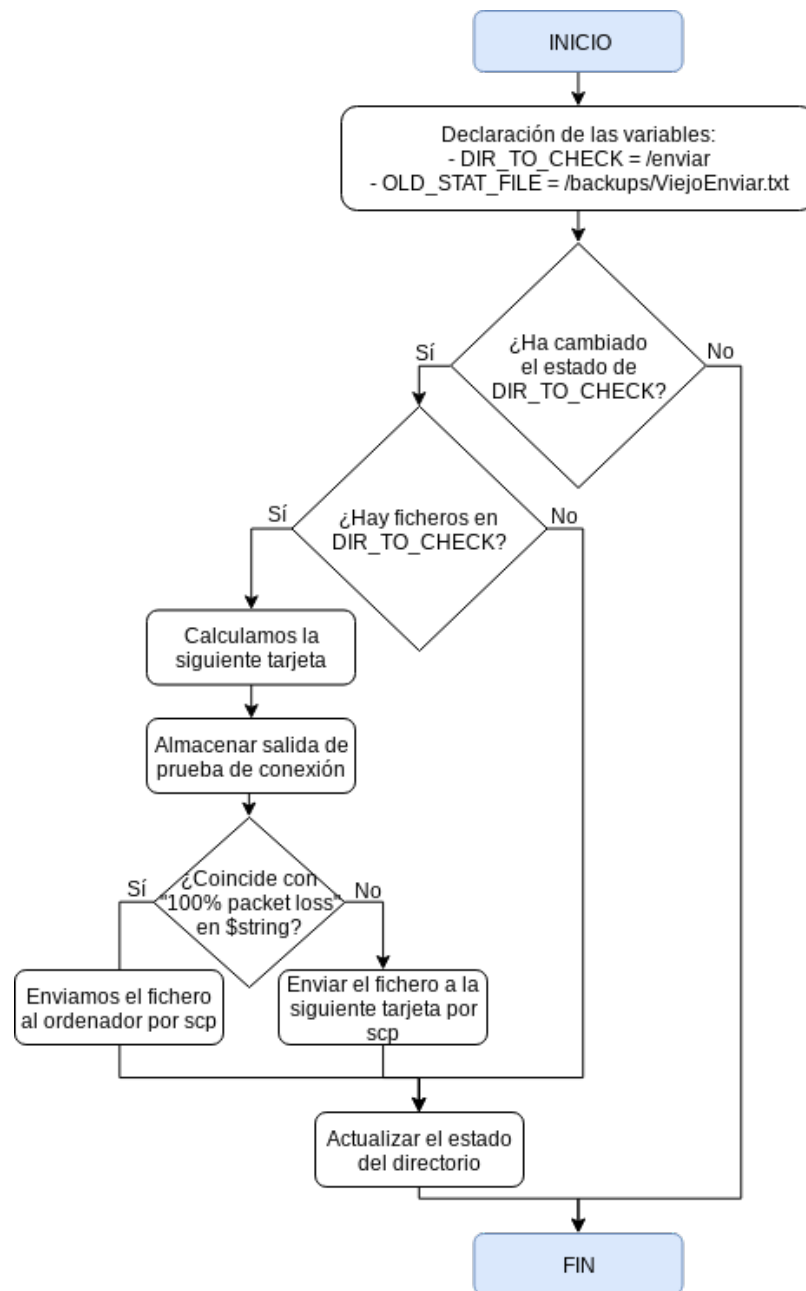


Figura 5: Diagrama de flujo de Diagramas/Enviando.sh.

### 3.5.2. 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=$(ls -t | head -1) #Obtenemos el fichero más reciente
29
30         actual=$(cat /etc/passwd | cut -d : -f1 | grep zybo)
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                 $fichero zybo@monitor:/home/zybo/Documento/Zybo
41         else
42             siguienteZybo=$siguiente
43             siguienteZybo+=@
44             siguienteZybo+= $siguiente
45             siguienteZybo+=:/home/
46             siguienteZybo+= $siguiente
47             siguienteZybo+=/ficheros/recibir
```

```

48         sshpass -p $siguiente scp -o StrictHostKeyChecking=no
           $fichero $siguienteZybo
49     fi
50 fi
51 echo $NEW_STAT > $OLD_STAT_FILE #Actualiza el estado del directorio
52 fi

```

Código de Enviando.sh.

### 3.6. Borrar.sh

Este script se encarga de borrar el contenido de los directorios ~/ficheros/recibir, ~/ficheros/desencriptar, ~/ficheros/trabajar y ~/ficheros/enviar de las tarjetas Zybo.

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

./Borrar.sh

#### 3.6.1. Diagrama de flujo



Figura 6: Diagrama de flujo de Borrar.sh.

### 3.6.2. Código

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

Código de Borrar.sh.