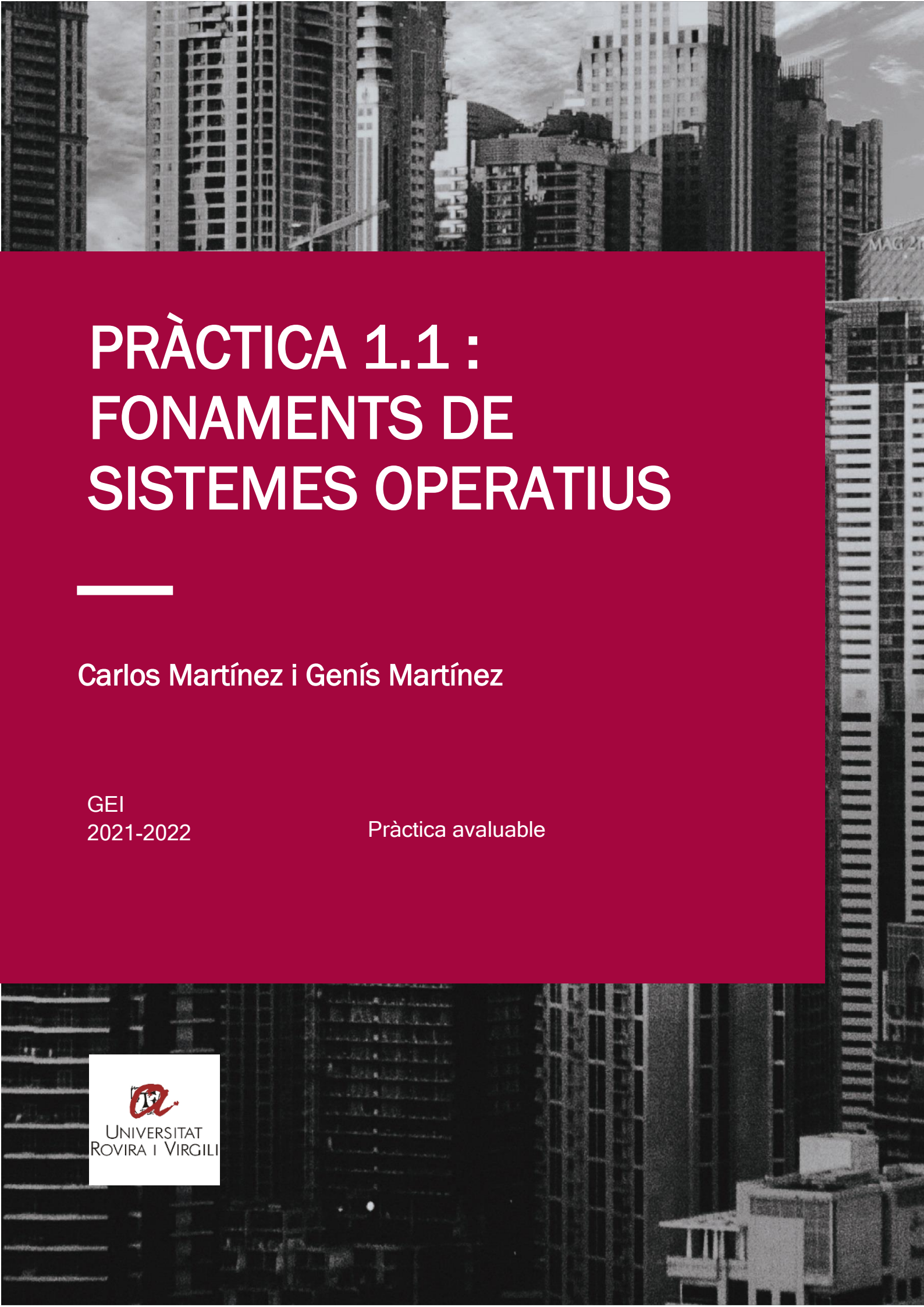


PRÀCTICA 1.1 : FONAMENTS DE SISTEMES OPERATIUS

Carlos Martínez i Genís Martínez

GEI
2021-2022

Pràctica avaluable

UNIVERSITAT ROVIRA I VIRGILI

Índex

1.- *Decisions de disseny*

Per a realitzar aquesta pràctica el primer que vam fer va ser repasar conceptes sobre “bash” explicats als primers laboratoris i investigar sobre la programació estructurada en funcions. Un cop realitzat això vam decidir dissenyar diverses funcions encarregades de subdividir les tasques que s’haurien de realitzar per al correcte funcionament de la pràctica. Aquestes són:

- **readFile** – llegeix les dades del fitxer
- **writeFile** – escriu les dades al fitxer
- **introducirRestantes** – indica quins elements falten per introduïr
- **printVariables** – llista els valors de les variables
- **comprobarValores** – comprova que els valors de les variables estiguin dins dels rangs correctes
- **comprobarFichero** – comprova que existeix el fitxer i trosseja les variables
- **readFromKeyboard** – segons les variables que el “getops” detecti que falten, les demana per teclat

Un cop encapsulades les tasques en aquestes funcions, a la part inferior de tot el codi fem les crides en l’ordre lògic per a que el programa funcioni correctament. Comprovem l’existència d’un arxiu anterior, comprovem els valors, introduïm els restants, etc...

2.- *Pla de proves*

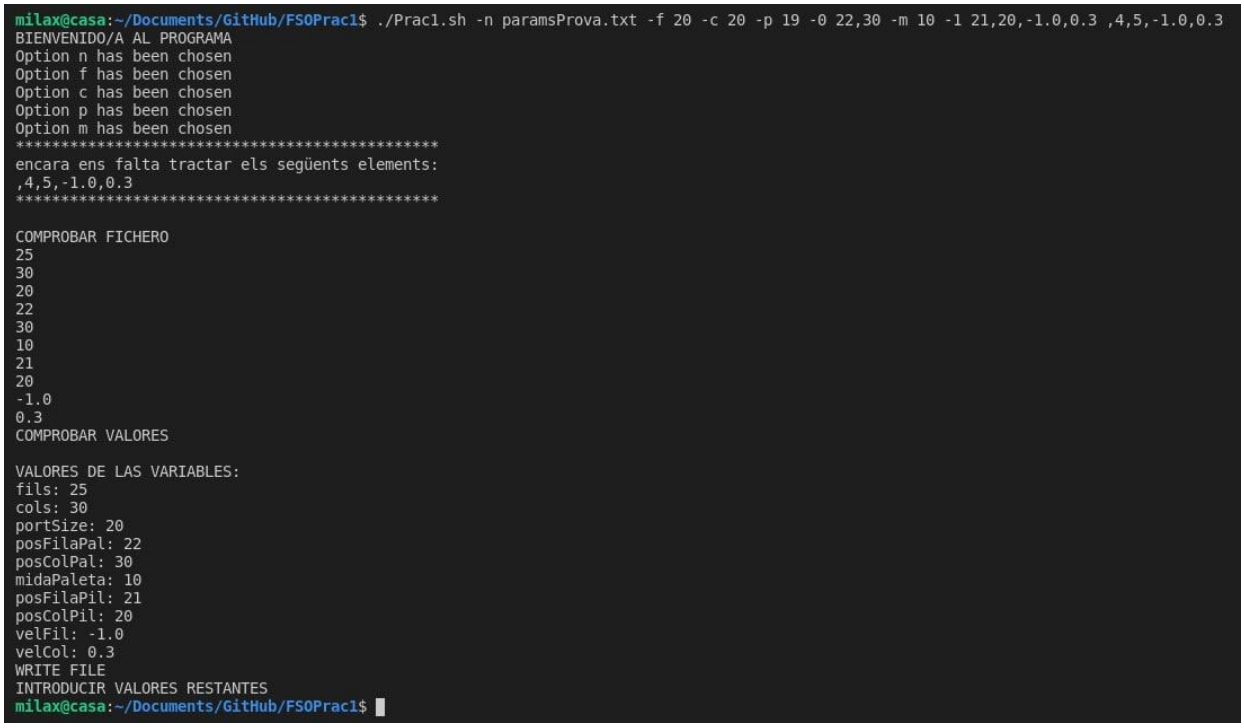
Per a comprovar que la nostra solució al problema és funcional, hem seguit la comanda d'exemple proporcionada al guió de la pràctica amb diferents valors. Per exemple:

./configuracio -n nomFit -f 25 -c 40 -p 20 -0 22,30 -m 10 -1 21,20,-1.0,0.3 ,4,5,-1.0,0.3

Ademés:

- Hem anat canviant els valors de les variables per comprovar que es guardaven correctament.
- Hem creat fitxers amb posicions buides per veure com es comportava el programa.
- Hem creat fitxers amb variables fora de rang.
- Hem introduït per comanda variables fora de rang.
- Hem introduït per comanda un nom de fitxer inexistent.

Prova d'exemple:



3.- *Codi font, conclusions i autoavaluació*

Codi font:

```
#! /bin/bash
fil=0
cols=0
portSize=0

minFil=10
maxFil=120

minCol=8
maxCol=36

minPort=8

minPort=8
temp=fil
posFilaPal=0
posColPal=0

posFilaPil=0
posColPil=0
velFil=0
velCol=0


Nflag=''
Fflag=''
Cflag=''
Pflag=''
OFlag=''
OneFlag=''
MFlag=''

found='false'
extraPilotes='false'
function readFile
{
    echo "READ FILE"
    #falta arreglar, sólo lee si hay 2 líneas en el archivo
    echo "Entra en 'readFile'"
    while IFS=' ' read -r fil cols portSize
    do
        echo "fils= $fil"
        echo "cols= $cols"
        echo "portSize= $portSize"
```

```
        exit
    done < $file
}

function writeFile
{
    echo "WRITE FILE"
    echo "$fil $cols $portSize" > $file
    echo "$posFilaPal $posColPal $midaPaleta" >> $file
    echo "$posFilaPil $posColPil $velFil $velCol" >>$file


}
function introducirRestantes
{
    echo "INTRODUCIR VALORES RESTANTES"
    if [ -z $fil ] || [ -z $cols ] || [ -z $portSize ] ; then
        read -p "Introduce de nuevo las filas (10,120): " fil
        read -p "Introduce de nuevo las columnas (10,36): " cols
        let maxS=fil-1
        read -p "Introduce de nuevo el tamaño de la portería (8, $maxS): " portSize
    fi

    if [ -z $posFilaPal ] || [ -z $posColPal ] || [ -z $midaPaleta ] ; then
        read -p "Introduce de nuevo posFilaPal (1,118): " posFilaPal
        read -p "Introduce de nuevo posColPal (2,35): " posColPal
        read -p "Introduce de nuevo midaPaleta (3,$maxS): " midaPaleta
    fi

    if [ -z $posFilaPil ] || [ -z $posColPil ] || [ -z $velCol ] || [ -z $velCol ];
then
        read -p "Introduce de nuevo posFilaPil (2,118): " posFilaPil
        read -p "Introduce de nuevo posColPil (2,35): " posColPil
        read -p "Introduce de nuevo velFil (-1.0-1.0): " velFil
        read -p "Introduce de nuevo velCol (-1.0-1.0): " velCol
    fi
}
function printVariables
{

    echo ""
    echo "VALORES DE LAS VARIABLES:"
    echo "fils: $fil"
    echo "cols: $cols"
    echo "portSize: $portSize"
    echo "posFilaPal: $posFilaPal"
    echo "posColPal: $posColPal"
    echo "midaPaleta: $midaPaleta"
    echo "posFilaPil: $posFilaPil"
    echo "posColPil: $posColPil"
    echo "velFil: $velFil"
```

```
    echo "velCol: $velCol"
}
function comprobarValores
{
    echo "COMPROBAR VALORES"
    if [ $fil -lt $minFil ] || [ $fil -gt $maxFil ] ; then
        read -p "Files incorrectes, torna a provar (10-120): " fil
    fi

    if [ $cols -lt $minCol ] || [ $cols -gt $maxCol ] ; then
        read -p "Columnes incorrectes, torna a provar (10-36): " cols
    fi
    let maxPort=fil-1
    if [ $portSize -lt $minPort ] || [ $portSize -gt $maxPort ] ; then
        read -p "Porteria incorrecta, torna a provar (8-numFiles-1): " portSize
    fi
    if [ $posFilaPal -lt 2 ] || [ $posFilaPal -gt 118 ] ; then
        read -p "posFilaPal incorrecta, torna a provar (2-118): " posFilaPal
    fi
    if [ $posColPal -lt 2 ] || [ $posColPal -gt 35 ] ; then
        read -p "posColPal incorrecta, torna a provar (2-35): " posColPal
    fi
    if [ $midaPaleta -lt 3 ] || [ $midaPaleta -gt $maxPort ] ; then
        read -p "Mida paleta incorrecta, torna a provar (3-numFil-1): " midaPaleta
    fi
    if [ $posFilaPil -lt 2 ] || [ $posFilaPil -gt 118 ] ; then
        read -p "posFilaPil incorrecta, torna a provar (2-118): " posFilaPil
    fi
    if [ $posColPil -lt 2 ] || [ $posColPil -gt 35 ] ; then
        read -p "posColPil incorrecta, torna a provar (2-35): " posColPil
    fi

    if (( $(echo "${velFil} > 1.0 || ${velFil} < -1.0" | bc -l ) )) ; then
        read -p "velFil incorrecta,  torna a provar (-1, 1): " velFil
    fi
    if (( $(echo "${velCol} > 1.0 || ${velCol} < -1.0" | bc -l ) )) ; then
        read -p "velCol incorrecta, torna a provar (-1, 1): " velCol
    fi
}

function comprobarfichero
{
    echo "COMPROBAR FICHERO"

    carlos='carlos'
    file='/home/milax/Documents/GitHub/FSOPrac1/params.txt'
    #file='/home/milax/FSOPrac1/params.txt'

    if [ -f $file ] && [ -e $file ]
    then
        num=0
```

```
        while read line; do
        if [ $num -eq 0 ]
        then
            fil=`echo $line | cut -d' ' -f1`
            echo "$fil"
            cols=`echo $line | cut -d' ' -f2`
            echo "$cols"
            portSize=`echo $line | cut -d' ' -f3`
            echo "$portSize"
        elif [ $num -eq 1 ]
        then
            posFilaPal=`echo $line | cut -d' ' -f1`
            echo "$posFilaPal"
            posColPal=`echo $line | cut -d' ' -f2`
            echo "$posColPal"
            midaPaleta=`echo $line | cut -d' ' -f3`
            echo "$midaPaleta"

        elif [ $num -eq 2 ]
        then
            posFilaPil=`echo $line | cut -d' ' -f1`
            echo "$posFilaPil"
            posColPil=`echo $line | cut -d' ' -f2`
            echo "$posColPil"
            velFil=`echo $line | cut -d' ' -f3`
            echo "$velFil"
            velCol=`echo $line | cut -d' ' -f4`
            echo "$velCol"


        elif [ $num -eq 3 ] && [ ! -z "$line" ]
        then
            extraPilotes='true'
            posFilaPil=`echo $line | cut -d' ' -f1`
            echo "$posFilaPil"

            posColPil=`echo $line | cut -d' ' -f2`
            echo "$posColPil"
            velFil=`echo $line | cut -d' ' -f3`
            echo "$velFil"
            velCol=`echo $line | cut -d' ' -f4`
            echo "$velCol"
        fi
        let num=num+1


        found='true'
        done < $file
        let maxPort=$fil-1

    else
        echo "No existe el archivo $file, procedemos a crearlo"
```

```
        touch $file #si no existe el fichero lo creamos

    fi
}


function readFromKeyboard
{
    echo "READ FROM KEYBOARD"
    if [ -z $Nflag ] ; then

        read -p "Introduce el nombre del archivo que quieres usar: " file
    fi

#FILAS
    if [ -z $Fflag ] ; then
        read -p "Introduce el numero de filas: " fil
    fi
        let maxPort=fil-1
        #rows output
        truncate -s-1 file
        #echo -n "$fil " > $file
#COLUMNAS
    if [ -z $Cflag ] ; then
        read -p "Introduce el numero de columnas: " cols
    fi
        #columns output//lo guardamos en el fichero $file
        truncate -s-1 file
        #echo -n "$cols " >> $file

#PORTERIA
    if [ -z $Pflag ] ; then
        read -p "Introduce la medida de la porteria: " portSize
    fi
        #output porteria// lo guardamos en el fichero $file
        truncate -s-1 file

        #echo -n "$portSize" >> $file
        comprobarValores
}


echo "BIENVENIDO/A AL PROGRAMA"
#primero de todo comprobamos que el fichero deseado existe
#si el fichero existe debemos comprobar que todos los parametros estan en regla
    #readFile
#en caso contrario pediremos los parametros que faltan porteclado al usuario
    #readFromKeyboard


while getopts 'n:f:c:p:m:0:1:' opcio; do
    case "${opcio}" in
```

```
        n) echo "Option n has been chosen" ;Nflag='true';file="${OPTARG}";;
        f) echo "Option f has been chosen";Fflag='true';fil=${OPTARG};;
        c) echo "Option c has been chosen";Cflag='true';cols=${OPTARG};;
        p) echo "Option p has been chosen";portSize=${OPTARG};Pflag='true';;
        m) echo "Option m has been chosen";midaPaleta=${OPTARG};MFlag='true';;
        0)IFS=',';read -a strarr <<<
"${OPTARG}";posFilaPal=${strarr[0]};posColPal=${strarr[1]};OFlag='true';;
        1) IFS=',';read -a split <<< "${OPTARG}";posFilaPil=${split[0]};
            posColPil=${split[1]};velFil=${split[2]};velCol=${split[3]};OneFlag='tr
ue';;
        #*) error "Unexpected option ${opcio}" ;;
    esac
done
shift $(($OPTIND - 1))
echo "************************************************"
printf "encara ens falta tractar els següents elements: %s\n$* \n"
echo "************************************************";echo""
comprobarfichero
comprobarValores
printVariables
writeFile

#readFromKeyboard
introducirRestantes
```

Conclusions:

Realitzant aquesta pràctica hem pogut entendre amb més precisió com funcionen els scripts i les comandes de la terminal. Durant la realització d'aquesta, ens hem trobat amb diferents problemes que, la majoria ja sabiem solucionar en altres llenguatges de programació, però haviem de descobrir com fer-ho en bash. Un d'ells ha sigut el fet de llegir i escriure fitxers de text. Un altre ha sigut separar els valors que llegim del fitxer. Ens va portar un temps també entendre com funcionava el "getops" i com utilitzar els flags al nostre favor per determinar quines variables faltaven per tractar. Coses que com bé hem comentat, sabíem fer en altres llenguatges però hem hagut d'apendre en bash en aquesta ocasió. Per altre banda hem hagut de repasar conceptes de C ja que al no ser un llenguatge orientat a objeces les funcions i el main del programa conviuen en el mateix fitxer, cosa que fa que en codis mitjanament llargs sigui una mica embolicat tot.

Autoavaluació:

Concepte a avaluar	Puntuació
Decisions de disseny (bash i python). Ús de funcions amb paràmetres. Documentació.	0,75
Disseny del Joc de Proves (bash i python). Documentació	0,75
Creació d'un fitxer amb opcions vàlides passades per línia d'ordres amb una pilota	0,35
Importar opcions d'un fitxer i sobreescriure algunes opcions passades per la línia d'ordres (amb una pilota)	0.1
Demanar opcions concretes a l'usuari si no s'han entrat per línia d'ordres	0,4
Demanar opcions concretes a l'usuari si estan fora del rang	0,5
Passar més d'una pilota per línia d'ordres, controlant màxim 9 pilotes	0,2
Què es fa si s'afegeixen pilotes i el fitxer existeix?	0,25
Tractament d'errors/excepcions	0,25
Validació dinàmica dels rangs de les variables	0,5
Implementació i ús correcte de les comandes de la bash	1,8
Implementació i ús correcte de les comandes python	1,6