Sistemas Operativos *Ano letivo 2018/2019*

Docente: Nuno Lau

Relatório das Ferramentas de descoberta do espaço ocupado em disco

Trabalho realizado por:

Tomás Batista, nº89296

João Dias, nº89236

Índice

Introdução

No âmbito da unidade curricular de Sistemas Operativos, foi-nos proposto a realização de um trabalho cujo objetivo é o desenvolvimento de scripts em bash que permitem descobrir o espaço ocupado em disco por ficheiros com determinadas características e que podem ser solicitados a ser apagados. É possivel visualizar o total do espaço ocupado em disco por todos os ficheiros que se encontram em todas as subdirectorias das directorias que lhe são passadas como argumento e eliminar da contabilização o espaço ocupado por ficheiros essenciais. Os scripts podem receber parâmetros diferentes que irão imprimir resultados conforme a função destinada a cada um.

Desenvolvimento

Sendo o objetivo deste trabalho prático o desenvolvimento de scripts que permitem a descoberta do espaço ocupado em disco, usámos dois scripts, o *totalspace.sh* e o *nespace.sh.*

O ***totalspace.sh*** e o **nespace.h** apenas diferem na opção -e. Ambos os scripts podem receber as opções -L x (indica x maiores ficheiros de entre todas as diretorias a serem impressos), -l x (indica x ficheiros dentro de cada diretoria a serem impressos), -n x (indica uma expressão regular x que é verificada com o nome dos ficheiros), -d x (indica uma data x de limite máximo de acesso aos ficheiros) e -r e -a que indica a maneira de aparecerem ordenados os conteúdos (r -> ordem contrária e a -> ordem alfabética). O **nespace.h** tem ainda a opção de -e x (em que x é um ficheiro com nomes de ficheiros que devem ser ignorados).

De modo a saber que opções o utilizador passa como argumento usamos a opção **getopts** que verifica também a correta utilização das mesmas. Usámos flags para determinar que opções estavam a ser usadas começando todas elas a 0 e iam incrementando correspondentemente ao número de vezes que eram passadas, assim, conseguimos detetar se o utilizador passava várias vezes a mesma opção. Usámos também verificações para saber se o utilizador passava números nas opções -L e -l.

Para ambos os scripts recorremos a uma função recursiva que durante a sua execução apenas guarda ficheiros que obtenham correspondência com as opções passadas e no fim são impressas de acordo com as opções de ordenação passadas. Essa verificação foi suportada com as flags e com vários tipos de condições, if's, quando queríamos combiner várias opções, combinávamos os if's de ambas as opções. Para tal testámos várias condições para saber que opções estavam a ser passadas. No **nespace.sh** tivemos de adicionar mais condições para testar flags e combinar todos os casos possíveis e também adicionar mais o if para a opção -e.

De modo a testar que ficheiros determinar com "NA", testamos se o ficheiro é readable, se não for todos os sizes passam a ser "NA".

Para lidarmos com espaços nos diretórios passados pelo utilizador usámos a opção IFS, que interpreta de maneira correta os espaços.

```
#GETOPTS WILL GET THE OPTIONS PASSED BY THE USER
while getopts ':n:f:l:d:L:ra' OPTION; do
    case "$OPTION" in
        n)
            #Increment the times that the user passed each option
            n_flag=$((n_flag + 1))
            #If user passes more than 1 time the options it exits and print an error message
            if [ $n_flag -ne 1 ]; then
                echo "ERRO: Apenas pode passar uma vez cada opcao"
                exit 2  #2=code error
            fi
            nvalue="$OPTARG"
            #If the argument passed to the option is not a number displays error and exit
            if ! [[ "$nvalue" =~ ^[0-9]+$ ]]
            then
                echo "ERRO: Passe um numero valido no argumento das opcoes"
                exit 3 #3=code error
            fi
        ;;
        l)
            l_flag=$((l_flag + 1))
            if [ $l_flag -ne 1 ]; then
                echo "ERRO: Apenas pode passar uma vez cada opcao"
                exit 2  #2=code error
            fi
            #If -l and -L are passed to the function, display an error message and exits the script
            if [[ $l_flag == 1 && $L_flag == 1 ]]; then
                printf "Not possible to combine -l and -L, try again please.\n"
                exit 1  #1=code error
            fi
            lvalue="$OPTARG"
            if ! [[ "$lvalue" =~ ^[0-9]+$ ]]
            then
                echo "ERRO: Passe um numero valido no argumento das opcoes"
                exit 3 #3=code error
            fi
        ;;
        d)
            d_flag=$((d_flag + 1))
            if [ $d_flag -ne 1 ]; then
                echo "ERRO: Apenas pode passar uma vez cada opcao"
                exit 2  #2=code error
            fi
            dvalue="$OPTARG"
            #This if checks if the -d option argument (a date) is true, $? -eq 0 means true
            #If it is true it convert the argument of -d to a date
            if [ "$d_flag" = 1 ]; then
                if [ $? -eq 0 ]; then
                    arg_date=$(date -d $dvalue "+%s")
                fi
            fi
        ;;
        L)
            L_flag=$((L_flag + 1))
            if [ $L_flag -ne 1 ]; then
                echo "ERRO: Apenas pode passar uma vez cada opcao"
                exit 2  #2=code error
            fi
            #If -l and -L are passed to the function, display an error message and exits the script
            if [[ $l_flag == 1 && $L_flag == 1 ]]; then
                printf "Not possible to combine -l and -L, try again please.\n"
                exit 1  #1=code error
            fi
            Lvalue="$OPTARG"
            if ! [[ "$Lvalue" =~ ^[0-9]+$ ]]
            then
                echo "ERRO: Passe um numero valido no argumento das opcoes"
                exit 3 #3=code error
            fi
        ;;
        r)
            r_flag=$((r_flag + 1))
            if [ $r_flag -ne 1 ]; then
                echo "ERRO: Apenas pode passar uma vez cada opcao"
                exit 2  #2=code error
            fi
        ;;
        a)
            a_flag=$((a_flag + 1))
            if [ $a_flag -ne 1 ]; then
                echo "ERRO: Apenas pode passar uma vez cada opcao"
                exit 2  #2=code error
            fi
        ;;
        \?)
            echo "$(basename $0) OPTIONS: [-n arg] [-l arg] [-d arg] [-L arg] [-r] [-a]" >&2
            exit 1
        ;;
    esac
done
shift "$((OPTIND - 1))"
```

Interpreta o input fornecido pelo utilizador

```
#N TRUE
if [[ "$d_flag" -eq 0 && "$n_flag" -eq 1 ]]; then
    base_name="$(basename "${entry}")"
    #If the name of the file contains (or it is totally equals) it stores
    #the file name and the directory path (and respective size)
    if [[ "$base_name" =~ ^$nvalue$ ]]; then
        files["${base_name}"]=$size
        string=$(dirname "$base_name")
        dirs["$string"]+=$size
        totalspace=$((totalspace + size))
    fi
fi
```

Verificação para a opção -n

```
#D TRUE
if [[ "$d_flag" -eq 1 && "$n_flag" -eq 0 ]]; then
    file_data=$(date -d "$file_data" +%s)
    #If the last modify on the file it is older than the date passed on the argument
    #it stores the file and the directory path (and respective size)
    if [ "$arg_date" -ge "$file_data" ]; then
        files["${entry}"]=$size
        string=$(dirname "$entry")
        dirs["$string"]+=$size
        totalspace=$((totalspace + size))
    fi
fi
```

Verificação para a opção -d

Sorting

As opções -r e -a apenas vão surtir efeito no final da execução do Código. Uma vez mais, com flags, determinamos qual dos arrays associativos usar (o dos files, o dos diretórios ou o dos diretórios para a opção -l) e assim correspondetemente imprimir determinado array e determinada maneira. Baseamo-nos na coluna a ordenar, qual a ordem e qual a maneira (alfabeticamente ou numericamente)

```bash
#NUMERICALLY
if [[ "$r_flag" = 0 && "$a_flag" = 0 ]]; then
    #With the directories passed on the arguments calls the recursive function
    #IFS leads with with spaces on the directories or files names
    IFS=$'\n'
    for item in "$@"; do
        walk "${item}"
    done
    unset IFS

    #If the -L flag is on it will print from the Files array
    if [ "$L_flag" = 1 ]; then
        for k in "${!files[@]}"; do
            echo ${files["${k}"]} ${k}
        done | sort -rn -k1 | head -${Lvalue}   #Sorts numerically by the sizes column
    fi
    #If the -L flag is off it will print from the directories array
    if [ "$L_flag" = 0 ]; then
        for item in ${!dirs[@]}; do
            echo ${dirs["${item}"]} "${item}"
        done | sort -rn -k1 #Sorts numerically by the sizes column
    fi
fi
```

Ordena numericamente pelo tamanho das colunas

```bash
#NUMERICALLY REVERSE
if [[ "$r_flag" = 1 && "$a_flag" = 0 ]]; then
    #With the directories passed on the arguments calls the recursive function
    #IFS leads with with spaces on the directories or files names
    IFS=$'\n'
    for item in $@; do
        walk "${item}"
    done
    unset IFS

    #If the -L flag is on it will print from the Files array
    if [ "$L_flag" = 1 ]; then
        for k in "${!files[@]}"; do
            echo ${files["${k}"]} ${k}
        done | sort -rn -k1 | head -${Lvalue} | sort -n -k1 #Sorts numerically by the sizes column in reverse order
    fi
    #If the -L flag is off it will print from the directories array
    if [ "$L_flag" = 0 ]; then
        for item in ${!dirs[@]}; do
            echo ${dirs["${item}"]} "${item}"
        done | sort -n -k1 #Sorts numerically by the sizes column in reverse order
    fi
fi
```

Ordena numericamente pelo tamanha das colunas de ordem inversa

```
#ALPHABETICALLY
if [[ "$r_flag" = 0 && "$a_flag" = 1 ]]; then
    #With the directories passed on the arguments calls the recursive function
    #IFS leads with with spaces on the directories or files names
    IFS=$'\n'
    for item in $@; do
        walk "${item}"
    done
    unset IFS

    #If the -L flag is on it will print from the Files array
    if [ "$L_flag" = 1 ]; then
        for k in "${!files[@]}"; do
            echo ${files["${k}"]} ${k}
        done | sort -rn -k1 | head -${Lvalue} | sort -r -k2 #Sorts alphabetically by the files column
    fi
    #If the -L flag is off it will print from the directories array
    if [ "$L_flag" = 0 ]; then
        for item in ${!dirs[@]}; do
            echo ${dirs["${item}"]} "${item}"
        done | sort -k2 #Sorts alphabetically by the directories column
    fi
fi
```

Ordena alfabeticamente pela coluna de ficheiros

```
#ALFABETICAMENTE PELOS PATHS REVERSE
if [[ "$r_flag" = 1 && "$a_flag" = 1 ]]; then
    #With the directories passed on the arguments calls the recursive function
    #IFS leads with with spaces on the directories or files names
    IFS=$'\n'
    for item in $@; do
        walk "${item}"
    done
    unset IFS

    #If the -L flag is on it will print from the Files array
    if [ "$L_flag" = 1 ]; then
        for k in "${!files[@]}"; do
            echo ${files["${k}"]} ${k}
        done | sort -rn -k1 | head -${Lvalue} | sort -k2 #Sorts alphabetically by the files column in reverse order
    fi
    #If the -L flag is off it will print from the directories array
    if [ "$L_flag" = 0 ]; then
        for item in ${!dirs[@]}; do
            echo ${dirs["${item}"]} "${item}"
        done | sort -r -k2 #Sorts alphabetically by the directories column in reverse order
    fi
fi
```

Ordena alfabeticamente pela coluna de ficheiros de ordem inversa

```
e)
    e_flag=$((e_flag + 1))
    if [ $e_flag -ne 1 ]; then
        echo "ERRO: Apenas pode passar uma vez cada opcao"
        exit 2  #2=code error
    fi
    evalue="$OPTARG"
    #This will put all the lines of file in the array_files
    if [ "$e_flag" = 1 ];then
        i=0
        array_files=()
        while read line_data; do
            array_files[i]="${line_data}"
            ((++i))
        done < "$evalue"
    fi
;;
```

Nova opção possível

```
#E TRUE
if [[ "$d_flag" -eq 0 && "$n_flag" -eq 0 && "$e_flag" -eq 1 ]]; then
    base_name="$(basename "${entry}")"
    #It traverse the array with the files not to consider
    for item in ${array_files[@]}; do
        #If any of the files on the file it is equal to the one
        #Passed by argument it does not store
        if ! [[ "$base_name" =~ ^$item$ ]]; then
            files["${entry}"]="NA"
            string=$(dirname "$entry")
            dirs["$string"]="NA"
            totalspace=$((totalspace + size))
        fi
    done
fi
```

Verificação para a opção -e

Conclusão

Considerando todos os requisitos impostos no enunciado para cada script, podemos dizer que o resultado foi alcançado, os scripts funcionam como pedido, excetuando o -l que imprime apenas para cada diretoria os x maiores ficheiros isoladamente. Também o nosso NA apresenta inconsistências.

Apesar disso, o *totalspace.sh* permite a visualização do espaço ocupado pelos ficheiros que lhe são passados como argumento e o *nespace.sh* permite a indicação de um ficheiro que contém uma lista de ficheiros essenciais.

Este projeto permitiu-nos conhecer melhor como é trabalhar com Bash, ficheiros e Linux.

```
    #If the -l flag is on it will print from the List_Dirs array
    declare -A l_dirs_array
    if [ "$l_flag" = 1 ]; then
        for item in ${!dirs[@]}; do
            total=0
            soma=$(builtin cd "$item" && ls -al | grep '^[-l]' | sort -nr -k5 | head -${lvalue} | awk '{ print $5 }')
            for i in $soma; do
                total=$((i+total))
            done
            l_dirs_array["$item"]=$total
        done
        for item in ${!l_dirs_array[@]}; do
            echo ${l_dirs_array["${item}"]} "${item}"
        done | sort -r -k2 #Sorts alphabetically by the directories column
    fi
fi
```

Solução alternativa para -l

Ferramentas de Suporte

- https://github.com/tomas99batista/Projeto1_SO
- http://code.ua.pt/projects/trabalho-pratico-1-so

Bibliografia

- www.stackoverflow.com