Trabajo de Diseño y Administración de Sistemas Operativos

Alumno: Michael Laudrup Luis González

PEC DyASO 20/21

Primera PED

Introducción

El objetivo principal de está práctica es mostrar la información estadística más básica que utiliza el sistema operativo Ubuntu, el cual es una distribución de Linux, tanto a nivel general (todo el sistema) como a nivel especifico (por procesos).

Además, la información a nivel especifico, es decir, por procesos, debe ser filtrada para solo mostrar datos de los 10 primeros procesos que usen mayor porcentaje de CPU.

A nivel general se muestra información como el número de procesos activos, la variación de uso de CPU en un segundo, memoria total, ocupada y libre.

A nivel proceso, se muestra el pid del proceso (número que lo identifica univocamente), el usuario que lo invoco, el comando que lo invoco, %CPU utilizada, %de memoria utilizada… entre otros datos que se obtienen principalmente de el sistemas de archivos, concretamente el subsistema de archivos /proc/ que tiene todo lo relativo a la gestión de procesos por parte del sistema operativo.

Implementación

Dado que consiste en manejo de datos es importante describir en que tipo de estructura de datos se han almacenado y procesado los datos, se ha hecho lo siguiente.

A nivel datos globales, se ha utilizado variables unicas. Sin embargo, a nivel de datos por proceso se han utilizado arrays, que no tienen vinculación directa a nivel código pero si a nivel conocimiento programador.

Los datos están almacenados en diferentes vectores cuyo indice-i es indicativo del mismo proceso, es decir, en los vectores que almacenan el pid del proceso, el nombre de usuario, porcentaje de uso de Cpu...etc. La posición i de cada vector hace referencia al mismo proceso.

Además, se ha modularizado mucho el código en funciones y se ha comentado todo el código utilizando variables muy explicitas que indican su cometido de manera intuitiva.

Otros aspectos a destacar son:

- C**alculo de porcentaje de CPU:** el procentaje de CPU se ha calculado siguiendo el siguiente procedimiento:

1. Se ha calculado el tiempo de usuario y el tiempo en modo nucleo de cada proceso (Está información están en los campos 14 y 15 del ficehro proc/num_pid/stat) y se ha sumado en un instante t, tammbién se obtiene el momento en segundos en el que se obtiene este dato (con el comando date).

2. Se ha esperado un segundo.

3. Se ha calculado el tiempo usuario y el tiempo en modo núcleo de cada proceso y se ha sumado en un instante t+1, también se obtiene el nomento en segundos en el que se obtiene estos datos.

4. se calcula la variación del uso de CPU tanto en modo usuario como en modo nucleo del proceso frente a la variación del tiempo en segundos.

Nota: Además de esto se ha aprovechado este método para calcular el uso de CPU a nivel global.

- **Obtención de los 10 procesos con mayor porcentaje de CPU.**
  - Para esto se han ido añadiendo a un fichero temporal, el indice i del proceso como primera columna y el porcentaje de CPU asociado como segunda columna, posteriormente se ha

ordenado este fichero con el comando “sort” atendiendo como criterio la segunda columna, quedando un fichero ordenado donde la primera columna indica el indice del vector y la segundo el porcentaje de CPU denominado fichero_temporal2.

- Finalmente, solo se imprimen los 10 primeros indices que están en la primera columna del fichero temporal resultante de la anterior operación

- **Calculo de tiempo de ejecución:**

1. Se calcula tiempo transcurrido desde que se inicio el sistema (campo 1 del fichero /proc/uptime)

2. Se calcula el momento en el que empieza a ejecutarse un proceso(campo 22 del fichero /proc/num_pid/stat) en tics de reloj y se divide entre 100 para pasarlo en segundos.

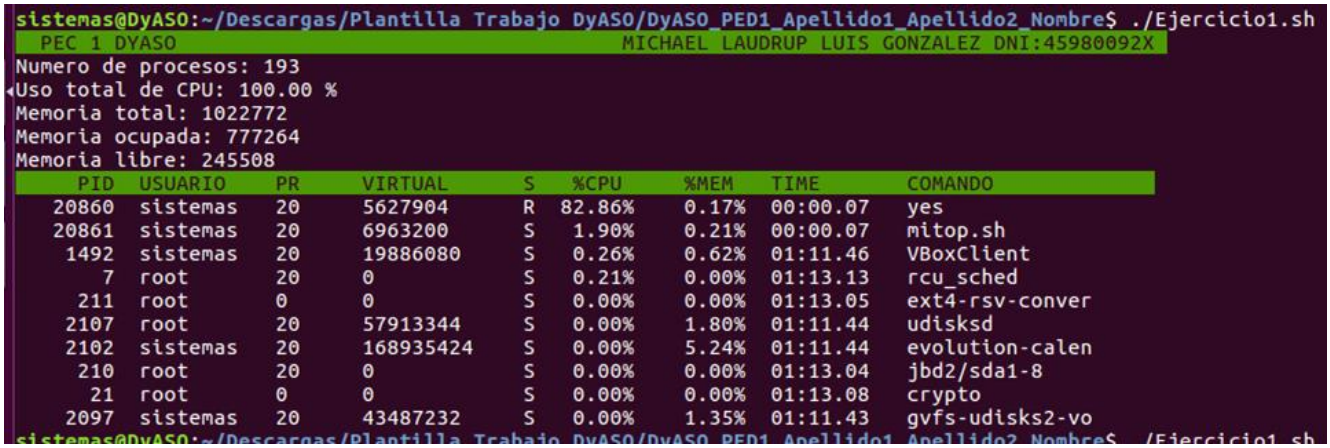
3. Se calcula diferencia entre el tiempo total en el que sistema ha estado activo menos tiempo en el que el proceso se inicio.

- **Resto de datos:**

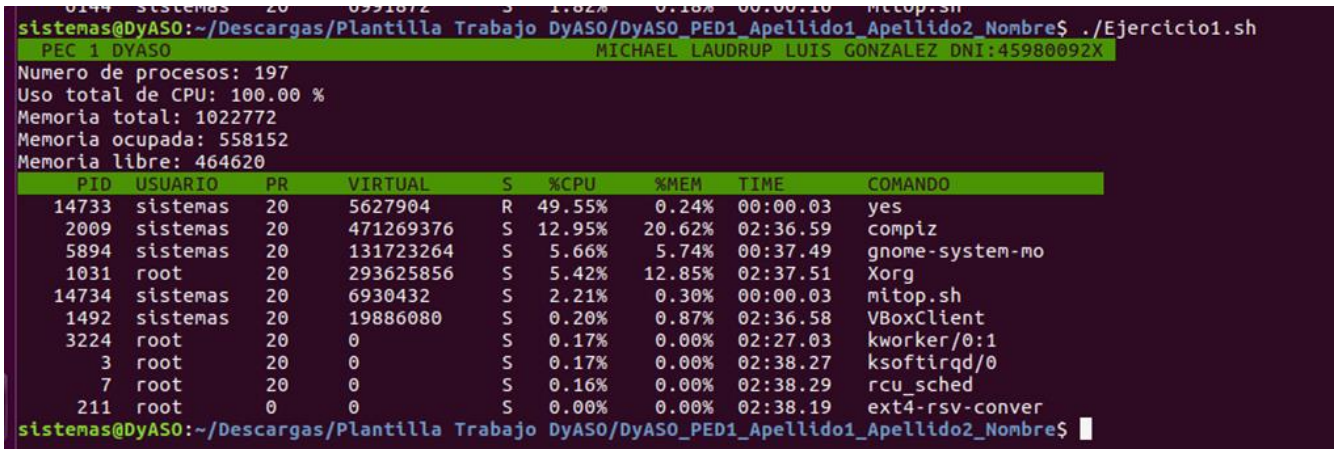
  - El resto de datos se obtienen realizando operaciones básicas sobre ficheros ubicados en la ruta /proc/pid_proceso/stat o /proc/stat, para la memoria /proc/meminfo, o /proc/[pid]/meminfo...etc. Dado que el código del archivo mitop.sh está muy bien comentado, se omite la especificación de cada campo para evitar ser redundante.

Ejecución de ejemplo

En este caso se puede aprecia que los datos tienen bastante sentido, dado que el proceso yes abarca un gran porcentaje de uso de CPU (lo cual, era de esperar), además, los siguientes procesos que más usan CPU son mitop.sh que justo está en ejecución en ese momento.



Otro ejemplo de ejecución:



En este caso el proceso “yes” no tiene tanto protagonismo pero porque se están teniendo en cuenta procesos potentes como el compiz (gestión de interfaz de ventanas del sistema operativo), xorg, mitop.sh, esto ha sucedido así porque mientras se ejecutaba este comoando continuaba navegando en otros programas, en detrimento del proceso “yes”.

Con respecto al resto de datos, considero que se puede apreciar que no siempre hay relacion directa en % de uso de CPU y % de Memoria, el tiempo es coherente con el tiempo que lleva encendido el sistema para algunos procesos, y el yes, por ejemplo, tiene una duración muy corta, lo cual, es lo esperado.

Código fuente

Resaltar que se quitan los comentarios del código para aumentar la limpieza del código

PORCENTAJE DE USO DE MEMORIA

```
calcularPorcentajeMemoriaUsada (){
  memory_size=$(awk '$1=="MemTotal:" {print $2}' /proc/meminfo)
  memory_free=$(awk '$1=="MemFree:" {print $2}' /proc/meminfo)
  memory_ocuped=$(expr $memory_size - $memory_free)
  memory_usage_pid=$(awk '{print $1}' /proc/"${vector_pids[$1]}"/statm)
  memory_usage_percent[$1]=$(echo "scale=10; $memory_usage_pid/$memory_ocuped*100" | bc )
}
```

TIEMPO DE EJECUCIÓN DEL PROCESO

```
calcularTiempoDeEjecucion(){
  start_time=$(awk '{print $22}' /proc/"${vector_pids[$1]}"/stat)
  start_time=$(echo "$start_time/$tics" | bc)
  uptime_system=$(awk '{print int($1)}' /proc/uptime)
  execution_time_seconds[$1]=$(echo "$uptime_system-$start_time" | bc)
}
```

PORCENTAJE DE USO DE CPU

```
function calcularPorcentajeUsoCPU (){
        for((j=1; j<= num_procesos;j++))
```

```
        do
                ruta_actual="/proc/"$(echo $procesos_PIDS | awk '{print $('$j')}')
                if [ -d $ruta_actual ]; then  #se comprueba que la ruta existe (algunos procesos son creados y eliminados por este propio script y no deben tenerse en cuenta)
                        total_time_t1[$j]=$(awk '{print $14+$15}' $ruta_actual/stat)  #suma de tiempo en modo usuario y modo nucleo de cada proceso en instante "t"
                        time_t1[$j]=$(date +%s.%N) #hora en el instante "t"
                fi
        done
total_cpu_t1=$(awk '$1=="cpu"{print $2+$3+$4+$5+$6+$7+$8}' /proc/stat) #suma de todo el uso de CPU en el instante t
        total_cpu_use_t1=$(awk '$1=="cpu"{print $2+$3+$4}' /proc/stat) #se quita el uso de CPU por procesos en modo usuario y modo supervisor en el instante t


        sleep 1    #espera de un segundo
        total_cpu_t2=$(awk '$1=="cpu"{print $2+$3+$4+$5+$6+$7+$8}' /proc/stat) #suma de todo el uso de CPU en el instante t
        total_cpu_use_t2=$(awk '$1=="cpu"{print $2+$3+$4}' /proc/stat) #se quita el uso de CPU por procesos en modo usuario y modo supervisor en el instante "t+1"
        cpu_usage=$(echo "scale=10;($total_cpu_use_t2-$total_cpu_use_t1)/($total_cpu_t2-$total_cpu_t1)*100" | bc) #variacion del uso de cpu en modo nucleo y usuario /variacion del uso CPu total
        for((k=1; k<= num_procesos;k++))
        do
          ruta_actual="/proc/"$(echo $procesos_PIDS | awk '{print $('$k')}')
          if [ -d $ruta_actual ]; then
                        total_time_t2[$k]=$(awk '{print $14+$15}' $ruta_actual/stat)
                        time_t2[$k]=$(date +%s.%N)
cpu_use_percent[$k]=$(echo "scale=4; ((${total_time_t2[$k]}-${total_time_t1[$k]})/$tics)/(${time_t2[$k]}-${time_t1[$k]}))*100" | bc)
                $(echo -e "$k ${cpu_use_percent[$k]}" >> /tmp/temporal_file)


                fi
        done
        sort -nrk2 /tmp/temporal_file > /tmp/temporal_file2
}
```

IMPRESION DE CABECERA

```
function imprimeCabecera(){
        echo -e "\e[7;32m  PEC 1 DYASO                        MICHAEL LAUDRUP LUIS GONZALEZ DNI:45980092X \e[0m"
```

```
        echo "Numero de procesos: $num_procesos"
    printf "%17s %5.2f %s \n" "Uso total de CPU:" "$cpu_usage" "%"
    memoriaTotal=$(awk '/MemTotal:/{print $2}' /proc/meminfo )
    memoriaLibre=$(awk '/MemFree:/{print $2}' /proc/meminfo )
        memoriaOcupada=$(echo "$memoriaTotal-$memoriaLibre" | bc)
    echo "Memoria total: $memoriaTotal"
    echo "Memoria ocupada: $memoriaOcupada"
    echo "Memoria libre: $memoriaLibre"
        echo -e "\e[7;32m    PID USUARIO    PR    VIRTUAL    S %CPU    %MEM TIME
COMANDO            \e[0m"
} #fin metodo de impresion de cabecera
```

IMPRESIÓN DE UN PROCESO SEGÚN SU PID

```
function imprimirProcesoIesimo(){
        ruta_actual="/proc/"$(echo $procesos_PIDS | awk '{print $('$1')}')
            if [ -d $ruta_actual ]; then
                        hours=$(echo ${execution_time_seconds[$1]}/3600 | bc)
                        minuts=$(echo "((${execution_time_seconds[$1]}%3600)/60)" | bc)
                        seconds=$(echo "((${execution_time_seconds[$1]}%3600)%60)" | bc)
                        printf "%8s  %-10s %-6s %-10s %3s  %5.2f%-1s   %5.2f%-
1s  %2.2d:%2.2d.%2.2d   %-25s \n" "${vector_pids[$1]}" "${vector_user_names[$1]}"
"${vector_priority[$1]}" "${vector_virtualSize[$1]}" "${vector_proc_status[$1]}"
"${cpu_use_percent[$1]}" "%" "${memory_usage_percent[$1]}" "%" "$hours" "$minuts"
"$seconds" "${vector_comand_invoked[$1]}"
            fi


}
```

PROGRAMA PRINCIPAL

```
#PROGRAMA PRINCIPAL
for((i=1; i<= $num_procesos;i++))
do
  ruta_actual="/proc/"$(echo $procesos_PIDS | awk '{print $('$i')}')  #en cada vuelta se actualiza la
ruta a /proc/[pid]
  if [ -d $ruta_actual ]; then  #si existe la ruta se procede a calcular sus datos (hay rutas numericas
```

que desaparecen y aparecen mientras se ejecuta el script y da problemas

```
        vector_pids[$i]=$(echo $procesos_PIDS | awk '{print $('$i')}')      # los vectores de datos se alinean segun un indice-i no segun pid, en este caso se asigna el pid

        vector_uids_num[$i]=$(awk '$1=="Uid:" {print $2}' $ruta_actual"/status" ) #asignacion de numero de usuario asociado al proceso de indice-i (NO PID)

        vector_user_names[$i]=$(awk -F:  '$3=="'${vector_uids_num[$i]}'"   {print $1}' "/"etc"/passwd") # se asocia num usuario con nombre de usuario en archivo passwd

      vector_priority[$i]=$(awk '{print $18}' $ruta_actual/stat) # se consulta la prioridad del proceso con indice-i

      vector_virtualSize[$i]=$(awk '{print $23}' $ruta_actual/stat) #se consulta la memoria virtual del proceso con indice-i

      vector_proc_status[$i]=$(awk '{print $3}' $ruta_actual/stat) #se consulta el estado del proceso con indice-i

      calcularPorcentajeMemoriaUsada $i #se calcula el procentaje de memoria usada del proceso con indice.-i

      calcularTiempoDeEjecucion $i #se calcula el tiempo de ejecucion del proceso con indice-i

      vector_comand_invoked[$i]=$(awk '{print $2}' $ruta_actual/stat | tr -d '()')   #se comprueba comando invocador de proceso con indice-i eliminando parentesis que envuelven comando

   fi

done

calcularPorcentajeUsoCPU #se calcula porcentaje de uso de CPU

imprimeCabecera #se imprime la cabecera con datos globales del sistema

for((v=1; v<=10;v++))

do

   index=$(awk 'NR == '$v' {print $1}' /tmp/temporal_file2)  #se imprimen solo los 10 primeros indices asociados a procesos que aparezcan en el archivo temporal_2 (tiene ordenado segun %cpu)

   imprimirProcesoIesimo $index #metodo que imprime proceso segun indice de vector de procesos

done


rm /tmp/temporal_file #se borran archivos residuales para que no influencien en posteriores ejecuciones

rm /tmp/temporal_file2

#FIN PROGRAMA PRINCIPAL
```

Bibliografía.

Enlace	Descripción
https://man7.org/linux/man-pages/man5/procfs.5.html	Descripción del susbsistema de ficheros de gestión de datos de procesos
https://francisconi.org/linux/comandos/sort	Ordenación de ficheros siguiendo criterio
https://www.softzone.es/programas/linux/ver-procesos-ram-cpu-linux/	Porcentaje de uso de CPU

http://congresos.nnb.unam.mx/TIB2014/sites/default/files/TIB2014/manual_awk.pdf	Manual de awk en pdf.
https://docs.microsoft.com/es-es/cpp/c-runtime-library/format-specification-syntax-printf-and-wprintf-functions?view=vs-2019	Formato de especificación de printf