Departament d'Enginyeria informàtica i Matemàtiques

Pràctica 1: Edició i Execució de Shell-Scripts en Python

Fonaments de Sistemes Operatius

EQUIP: Marc Cabré Guinovart, Aleix Mariné i Tena

ASSIGNATURA: **Fonaments de Sistemes Operatius**

DATA: 24 / Març / 2017

Índex

Enunciat

En aquesta pràctica es demana que es desenvolupi una aplicaió que permeti editar i executar shell-scripts. En el nostre cas, s'ha implementat l'aplicació amb totes les parts opcionals, de manera que en aquest apartat es descriurà l'aplicació amb aquests apartats.

Eines a Utilitzar

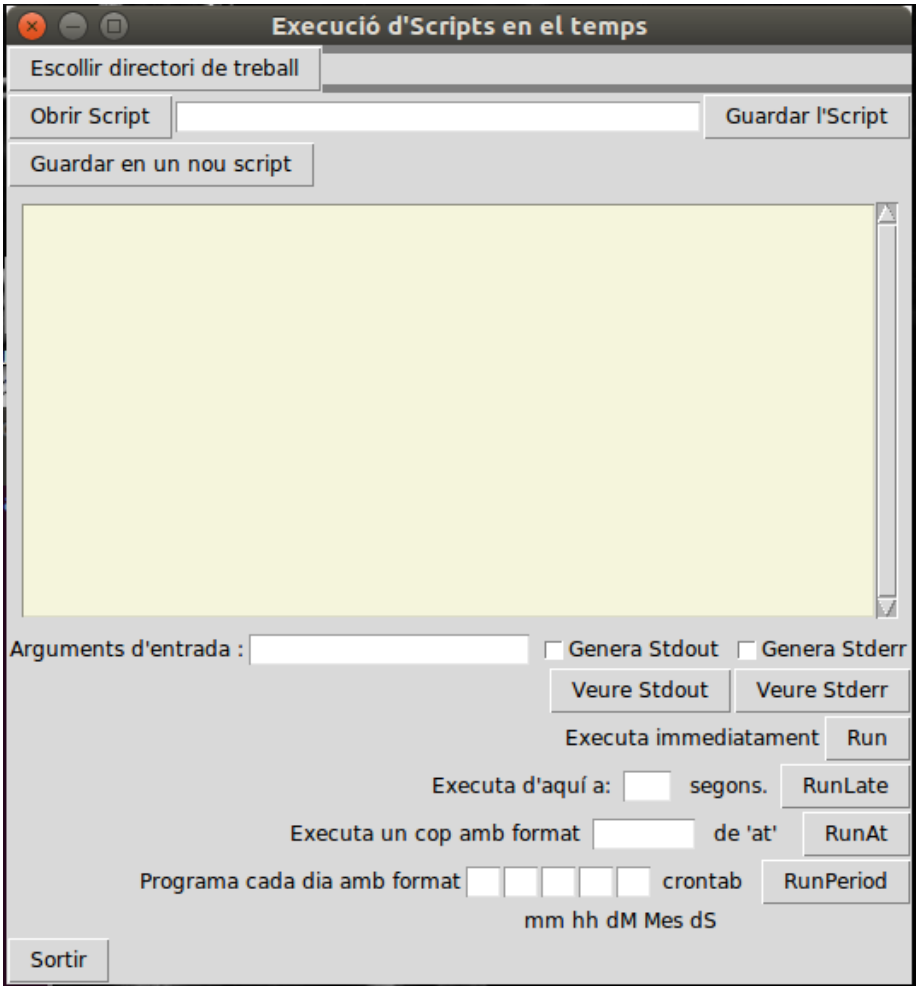
Es demana que per a desenvolupar l'aplicació s'utilitzi Python com a llenguatge de programació i que la Interfícies Gràfica es desenvolupi utilitzant la llibreria tkinter integrada en el llenguatge.

Entorn de Validació

Per la validació del correcte funcionament de la pràctica s'utilitzarà un sistema idèntic al instal·lat a l'aula (Milax). En el nostre cas hem utilitzat instal·lacions pròpies d'Ubuntu per a desenvolupar la pràctica i posteriorment hem validat el correcte funcionament amb la màquina virtual que hi ha al moodle de l'assignatura.

Funcions Implementades

S'ha implementat la següent interfície:



1. **Escollir directori de Treball:** Demana a l'usuari en quin directori base vol treballar. El directori de treball queda seleccionat i escrit en el camp de text corresponent.
2. **Obrir Script:** Demana a l'usuari que trii un fitxer que es carregarà en pantalla. El nom del fitxer es mostra per pantalla al camp corresponent.
3. **Guardar Script:** Permet a l'usuari guardar les modificacions fetes des del programa al script prèviament obert per tal de poder-ho executar després.
4. **Guardar en un nou Script:** Permet a l'usuari guardar un script quan és nou i no s'ha obert de cap fitxer. Si l'haviem obert d'un fitxer avisa a l'usuari que ja existeix.
5. **Generar Stdout:** Genera la sortida de l'script executat en un fitxer que té per nom el nom del fitxer i per extensió .out.
6. **Generar Stderr:** Genera la sortida de l'script executat en un fitxer que té per nom el nom del fitxer i per extensió .err
7. **Veure Stdout:** Mostra una pantalla nova amb la informació del Stdout de l'últim script executat.
8. **Veure Stderr:** Mostra una pantalla nova amb la informació del Stdout de l'últim script executat.
9. **Executar Inmediatament:** Executa l'Script que tenim obert. Si tenim seleccionat Generar Stdout i/o stderr, genera els fitxers corresponents.
10. **Execució retardada:** Executa l'Script que tenim obert amb els segons de retard que indica l'usuari.
11. **Execució amb At:** Programa l'script que tenim obert per a que s'executi quan decideix l'usuari al introduir el format d'arguments de la comanda At. L'String At el té per defecte l'aplicació i no cal que sigui introduït per l'usuari.
12. **Execució amb Crontab:** Programa l'execució de l'script mitjançant la comanda crontab.

Pseudo codi i Implementació

Finestra principal mainw:

S'han utilitzat diversos *frames* per a anar agrupant els botons segons el model d'interfície que ens donava l'enunciat. S'ha utilitzat el mètode *pack* dels components per a anar-los alineant. Els components gràfics que s'han utilitzat que no estan a les transparències de classe són el tk.scrolledText.

Degut a la gran quantitat de components gràfics s'han anat donant noms curts i genèrics que segueixen una determinada numeració. La numeració va en ordre descendent excepte en les parts gràfiques de la part opcional, ja que aquestes es van fer després.

A continuació es mostra el pseudocodi de la implementació de cada botó. S'han eliminat les parts del codi que no formen part realment de l'algoritme, com l'assignació dels títols.

Escull directori

```
acció EscullDirectori():
        booleà existeix=cert;
        Cadena directori=dialeg.preguntadirectori(existeix)        #Obtenim directori de treball
        l1.posartext(directori) # Assignem el directori al label
```

Obrim un diáleg per a que el usuari seleccioni el directori de treball. Amb existeix=cert ens assegurem que el directori existeixi i no faci falta implementar excepcions.

Obrir Script

```
acció ObrirScript():
        fitxer file=dialeg.seleccionafitxer(directoriinicial=directori, fitxersacceptats=fitxersd'scripting])
        si file llavors      # Si el fitxer no està buit
                intenta
                        nomf = obtenirruta(file)      # Obtenim el nom del fitxer
                        e1.neteja()                   # Borrem el contingut de l'entry
                        e1.escriu(nomf)               # I li assiggnem el nom del fitxer
                        fitxer f = obrir (file, 'r')  # mode lectura
                        Cadena content=f.read()       # Llegim tot el fitxer i l'emmagatzemem
                        st.neteja()                   # Fem un clean del text
```

```
                st.escriu(content)          # Insertem el contingut del fitxer
                f.tancar()                  # Tanquem el fitxer
            capta excepció errorE/S:
                mostramissatge="Error E/S") # Si captem una excepció E/S mostrem el missatge
```

Obrim un diàleg per a seleccionar el fitxer. Es força a que aquest fitxer sigui del tipus script bash (*.sh). Si l'usuari finalment no selecciona cap fitxer el si condicional ens evita entrar al cos del codi; sinó llegim el contingut de fitxer i el passem al nostre scrolledtext. La variable file es global i s'inicialitza amb 0. Això es té en compte en la següent funció. Si hi ha cap problema E/S el programa podrà manegar la excepció.

Guardar Script

```
acció GuardarScript():
    si file!=0 llavors     # Si no em carregat fitxers..
        intentar:
            fitxer f = obrir(file, 'w')
            f.neteja()
            Cadena text = st.obtenirtext() # Obtenim el text del text box
            f.escriure(text)          # Guardem el text al fitxer
            f.tancar()                # Tanquem fitxer
        capta Error E/S:
            dialeg.mostraerror()
    sino
        dialeg.mostraerror(«Cal carregar un fitxer abans»)
```

Si no hi ha cap fitxer obert (es comprova mirant si la variable file segueix sent 0) mostrem error. Sinó entrem al cos del codi. Obrim el fitxer i borrem el contingut. Obtenim el text del scrolledtext i ho escrivim al fitxer, després tanquem el fitxer. Si hi ha cap problema E/S el programa podrà manegar la excepció.

Guardar nou Script

```
acció GuardarNouScript():
    intenta
        fitxer f=dialeg.guardar(extensió='.sh', directoriinicial=directori)
        si f llavors:                 # Si l'usuari ha escollit fitxer
            Cadena text=st.obtenirtext()
```

```
                    f.escriure(text)
                    f.tancar()
        capta exepció E/S:
                    dialeg.mostraerror()
```

Obrim un diàleg que serà el que ho faci tot. Li diem que el fitxer ha de ser obligatòriament amb extensió .sh. Això no s'especifica a l'enunciat, ha sigut una decisió de disseny, ja que sinó l'usuari no pot obrir els scripts que ell mateix guarda. Si l'usuari no selecciona fitxer llavors el si evita que entrem en el cos del codi. Sinó obtenim el text del scrolledtext i ho guardem dins del fitxer que hem obtingut del diàleg. Finalment tanquem el fitxer.

Si hi ha cap problema E/S el programa podrà manegar la excepció.

Veure Stdout/Veure Stderr

Les funcions són anàlogues, així que només posarem el pseudocodi d'una.

```
Acció VeureStderr():
        dialeg r = nou dialeg(stdout)
        mostra(r)
```

Fa una crida a la classe diàleg, que es la següent:

```
classe dialeg(stdout)
    ScrolledText stout = nou ScrolledText()
    stout.neteja()  # Fem un clean del text
    stout.escriure(stdouttxt)  # Inserim el contingut del stdout
    botó b = botó(text="Close", commanda=tancardialeg())
```

Aquesta funció rep la cadena stdout, que es genera en la funció run now. Crea un nou diàleg. El diàleg esta composat per un scrolled text al que se li escriu la cadena stdouttxt i un botó per a tancar la finestra.

Executa Ara

```
acció RunNow():
        Cadena ubicacio="/tmp/tmp.sh"
        Cadena arguments = ubicacio + " " + llegir.args()
        Cadena com = st.llegir() # Obtenim el text del fitxer
```

```
fitxer f = obrir(ubicacio, 'w')
f.netejar()
f.escriure(com)
f.tancar()
aplica_permisos(fitxer=ubicacio, permis=execucio)          # Donem permisos d'execució
stdouttxt, stderrtxt  = executa_script(ubicacio) # Obtenim les sortides
si checkboxstdout.marcat()
        si nomf!=0
                f = obrir (nomf+".out", 'w+')
                f.neteja()
                f.escriure(stdouttxt)
                f.tancar()
        sino
                f = obrir ("script.out", 'w')
                f.neteja()
                f.escriure(stdouttxt)
                f.tancar()
si checkboxstderr.marcat():
        si nomf!=0
                f = obrir (nomf+".out", 'w+')
                f.neteja()
                f.escriure(stdouttxt)
                f.tancar()
        sino
                f = obrir ("script.out", 'w')
                f.neteja()
                f.escriure(stdouttxt)
                f.tancar()
```

Es crea la cadena arguments que conté la ruta a l'script i els arguments de l'entry box.

S'ha pres la decisió de disseny de guardar el text escrit en el textbox en la carpeta tmp. Això fa que si l'usuari obre el programa i es dedica a teclejar codi el pugui executar, de igual manera que si obre un fitxer i introdueix canvis no guardats, pugui provar l'script abans de guardar. Això resulta més útil ja que no està limitat a que l'usuari premi el botó guardar.

Obtenim la el text de comandes i el escrivim al fitxer tmp.sh. Tanquem el fitxer i li donem permisos d'execució. L'executem i obtenim les sortides. Segons si l'usuari ha marcat els

checkboxes del stdout o stderr crearà els fitxers de sortida corresponents. En el cas de que l'usuari no hagi obert cap fitxer, se li donarà un nom genèric. Abans d'escriure s'esborrarà el contingut del fitxer per a evitar acumulació de sortides.

Executa Tard

```
acció RunLate():
        intenta
            int rnd = rand()
            int segons = e4.llegir()
            si (segons > 0) llavors:
                    Cadena ubicacio="/tmp/" + rnd + "Late.sh"
                    Cadena arguments = "sleep " + segons + " && " + ubicacio + " " + e2.llegir()
                   Cadena com = st.llegir()  # Obtenim el text del fitxer
                    Fitxer f = obrir(ubicacio, 'w')  # Obrim el fitxer en mode escriptura
                    f.escriure(com)  # Escrivim el script
                    f.tancar()  # Tanquem el fitxer
                   aplica_permisos(execució, ubicacio)  # Donem permisos d'execució
                   executa(arguments)
            sino:
                    diàleg.mostraerror()
      excepció:
            diàleg.mostraerror("Error E/S")
```

La funció anterior funciona de la mateixa manera que executa ara a la part estàndard d'execució, però amb la variació que ha d'obtenir el paràmetre dels segons del e2 (entry). Per a utilitzar el retard utilitzem la comanda sleep seguida de la execució del script. Això s'executarà en segon pla, de manera que no bloquejarà el programa.

Executa A les

```
Acció RunAt():
  intenta:
        int rnd = rand()
        Cadena commands = e7.llegir()
        si !commands.buit() llavors:
                diàleg.mostrarerror()
```

```
        sino:
                si directori==0 llavors:
                        Cadena ubicacio = obtenirubicacioscript()
                sino:
                        Cadena ubicacio=directori
        fitxer=ubicacio+rnd+”At.sh"
        arguments = "at " + commands + " -f " + "\"" + fitxer + "\"" + e2.get() + ""        com = st.get(1.0, END)
# Obtenim el text del fitxer
        fitxer f = obrir(fitxer, 'w')  # Obrim el fitxer en mode escriptura
        f.netejar()
        f.escriure(com)  # Escrivim el script
        f.tancar()  # Tanquem el fitxer
        aplicapermisos (execució, ubicació)
        out, err = executa(arguments)
                si !err.buit():
                        diàleg.mostraerror()
capta excepcio
        diàleg.mostraerror()
```

Funció periòdicament

```
accio RunPeriod():
        rnd = rand()
        com = st.llegir())
        si directori==0 llavors:
                        Cadena ubicacio = obtenirubicacioscript()
                sino:
                        Cadena ubicacio=directori
        fitxer=ubicacio + rnd + "Period.sh"
        booleà correcte = true
        #Per a tots els camps...
        dS = e5.llegir()
        si dS.buit():
                correcte = 0
        ...                     #... procedim igual
        si correcte == 1 llavors
                fitxer f = obrir(fitxer , 'w')
                f.netejar()
```

```
                f.escriure(com)
                f.tancar()
                donempermisos(execució, fitxer)
                fitxer s_file = obrir(ubicacio + "/crontask", 'w')
                s_file.escriure(mm + " " + hh + " " + dM + " " + Mes + " " + dS + " " +fitxerc + "\n")
                s_file.tancar()
                out, err  = executa("crontab " + ubicacio + "/crontask)
                si !err.buit():
                        diàleg.mostraerror()
sino:
                        diàleg.mostraerror()
```

Decidim quin serà el nostre directori de treball segons si l'usuari ha triat o no el directori. Si no l'ha triat agafarem com a directori de treball el de la ruta al nostre script.

En un fitxer escrivim les comandes com fèiem a la funció executa ara i li donem permisos d'execució. Després, llegim tots els camps de crontab i els acumulem a un string, donantli el format adequat. També afegim al final la ruta a l'script que volem executar. Escrivim al fitxer aquest string. Llavors fem una trucada al sistema a la comanda crontab i li passem la ruta al fitxer on tenim el string en format crontab. Finalment mirem la sortida, si el format esta correcte no hi haurà cap error, i no entrarem al if, sinó hi entrarem mostrant una advertència.

Llibreries Extres

ScrolledText

Referència: https://docs.python.org/2/library/scrolledtext.html

Scrolledtext és un mòdul que implementa un textbox multi-linia amb scroll bar vertical. Per implementar-lo hem utilitzat els següents mètodes:

```
tkst.ScrolledText(
  master=top,  # incloem a la finestra fout aquest objecte
  wrap='word',  # Fem que el text sigui de paraules (?)
  width=60,  # carácters per fila
  height=12,  # linies de text
  bg='beige'  # color de fons
)
stout.pack()
```

File Dialog

Referències:

- http://tkinter.unpythonic.net/wiki/tkFileDialog
- http://stackoverflow.com/questions/9319317/quick-and-easy-file-dialog-in-python

Implementa un mòdul que permet de manera fàcil preguntar a l'usuari el Path d'un fitxer amb el que després podrem treballar, o en aquest cas, editar.

text

Referència: https://www.tutorialspoint.com/python/tk_text.htm

Mòdul de la llibreria Tk per a mostrar un Label per pantalla

Files

Referències:

- https://docs.python.org/2/tutorial/inputoutput.html
- http://stackoverflow.com/questions/1466000/python-open-built-in-function-difference-between-modes-a-a-w-w-and-r

Llibreria de python que permet obrir, editar, llegir i tancar un fitxer fàcilment. El segon enllaça de les referències trobem els modes d'obrir un fitxer.

run

Referència: https://docs.python.org/2/library/subprocess.html

Subprocess és un mòdul que ens permet iniciar nous processos i connectar les seves outputs a Pipes del nostre programa per tal de poder analitzar les sortides estandard i d'error.

En el nostre cas hem tret molt de profit de la funció Popen implementada en aquest mòdul que ens permet molta flexibilitat a l'hora d'haver de manegar alguns casos

Els benefici principal d'aquesta funció és que executa la comanda en segon plà (en una nova tasca independent) de manera que no afecta a l'execució de la nostra aplicació, aconseguint així que en l'execució retardada, l'usuari no hagi d'esperar a que s'acabi l'execució amb la pantalla congelada.

A més hem redirigit l'output d'execució pitjançant Pipes de manera que amb l'execució de la funció communicate() sobre l'objecte retornat de la comanda popen, podem llegir lestàndard output i l'error output.

Exemple:

```
rc = subprocess.Popen(  # Cridem al modul subprocess
    arguments, # Li passem les comandes (execució + parametres)
    shell=True,  # Pel tipus de crida aquest ha de ser true
    stdout=PIPE, # Per a poder obtenir stdout
    stderr=PIPE)  # Per a poder obtenir stderr
    stdouttxt, stderrtxt = rc.communicate()    # Obtenim les sortides
```

crontab

Referència: http://blog.desdelinux.net/cron-crontab-explicados/#

Crontab és una llista de comandes amb una sèrie de paràmetres que indiquen en quin moment s'han d'executar. Crontab s'executa per cada usuari independentment i per a l'usuari root, de manera que cada usuari es pot definir les seves pròpies tasques.

Aquesta llista de tasques és revisada periòdicament per la comanda cron, un dimoni que si detecta que al revisar la llista de tasques alguna té els paràmetres de temps que coincideixen amb els d'aquell moment, executa la tasca.

Crontab a més a més també comprova el correcte format de la comanda que s'ha introduit.

Per a realitzar proves des de la consola de comandes hem utilitzat:

- Editar: crontab -e
- Esborrar tot: crontab -r
- Llistar: crontab -l

El format per al fitxer crontab és el següent

m h dom mon dow user command

on:

- m és el minut
- h és la hora
- dom és el dia del mes
- user és l'usuari que executarà la comanda
- command és la comanda a executar.

Exemple:

execució a les 10h 5m:

5 10 * * * usuario /home/usuario/scripts/actualizar.sh

execució cada minut:

* * * * * usuario /home/usuario/scripts/actualizar.sh

Cal afegir que per la manera que tenim de utilitzar els fitxers de crontab no necessitem posar l'usuari.

Joc de Proves

El Joc de proves en aquest cas s'ha realitzat en gran mesura manualment per tal de validar el correcte funcionament de la interfície gràfica:

- S’ha validat que el directori de treball es pot canviar i queda sempre escrit i desat en la memòria l’últim seleccionat.
- S’ha validat que el directori Home es pot canviar i queda sempre escrit i desat en la memòria l’últim seleccionat i que es carrega correctament el contingut del fitxer a la pantalla scrollText.
- S’ha validat que quan hem carregat un fitxer el podem desar amb els botons de desar, pero que no podem desar un fitxer nou amb el botó desar si no s’ha obert un fitxer previament.
- S’ha comprovat que els Scrolls funcionen correctament.
- S’ha validat l’execució del codi que tenim en pantalla amb el botó d’executar inmediatamnet i que el stdout i stderr es mostra correctament a les pantalles corresponents així com la generació dels fitxers de sortida estàndard i sortida d’error.
- S’ha validat que l’execució retardada no afecta a la usabilitat del programa i que s’executa correctament emb el marge de temps esperat.
- S’ha comprovat que la comanda At funciona correctament i que retorna els errors com s’espera. En cas de que la comanda no sigui ben reconeguda pel sistema mostrem d’stderr de l’execució de la comanda At a l’usuari amb un pop-up.
- S’ha comprovat que els paràmetres crontab s’accepten correctament i que les tasques es llisten al fitxer amb qualsevol combinació correcta. A més s’ha comprovat que si es posa una execució periòdica aquesta s’executa correctament totes les vegades. També s’ha comprovat que la nova comanda no elimina la anterior.