Budapesti Műszaki és Gazdaságtudományi Egyetem
Intelligens rendszerfelügyelet (VIMIA370)

Konfigurációkezelés

Házi feladat

2011. április 5.

1 Bevezető

Házi feladatomban megadott számítógépek hálózati eszközeit kell felderítenem, és azok részletes adatait a felhasználó által meghatározott kimenetre kiírni.

1.1 A házi feladat célja

Feladatomban egy gyors, áttekintő képet kell készítenem a gépeink fő hálózati beállításairól. Ehhez egy olyan Bash scriptet kell írnom, amely paraméterként megkapja egy CSV fájl elérési útját, majd a CSV fájlban felsorolt gépeken futó CIMOM-okat sorban lekérdezi CIM-XML segítségével, és mindegyikről kigyűjti a következő információkat. Fel kell sorolni, hogy milyen hálózati interfészek vannak beállítva rajtuk, és azok milyen fizikai címmel rendelkeznek, majd minden hálózati interfészhez meg kell jeleníteni a hozzá tartozó IP paramétereket (IP cím, alhálózati maszk). Az eredményt egy egyszerű szöveges jelentésben kell prezentálnom, ahol minden géphez tartozik egy interfész táblázat. Ennek sorai tartalmazzák az interfész nevét és fizikai címét, valamint az IP címet és alhálózati maszkot.

2 A szkript elkészítése

2.1 A paraméterek ellenőrzése

A program *kettő*, kötelezően megadandó *paramétert* vár. Mindkettő egy-egy fájl elérési útvonala. Az első paraméter a *bemeneti csv-fájl* útvonalát jelöli. Először azt ellenőrzöm, hogy a felhasználó megadta-e ezt a paramétert:

```
if [ -z $1 ]; then
       echo "Missing input file parameter";
       exit 1;
```

Ha hiányzik a paraméter, akkor a szkript a kimeneten tájékoztatja a felhasználót, majd 1-es hibakóddal kilép.

Ha a paramétert meg van adva, akkor ellenőrzöm, hogy a fájl valóban létezik-e:

```
elif [ ! -f $1 ]; then
       echo "Input file not exists";
       exit 3;
```

Ha a fájl nem létezik, akkor erről a program a kimeneten tájékoztatja a felhasználót, és 3-as hibakóddal leáll a futása.

Ezután következhet a második paraméter. Ez a *szöveges kimeneti fájl* útvonalát specifikálja. A paraméter meglétének ellenőrzése:

```
elif [ -z $2 ]; then
       echo "Missing output file parameter";
       exit 2;
```

Ha hiányzik a paraméter, akkor a szkript a kimeneten tájékoztatja a felhasználót, majd 2-es hibakóddal kilép

Ha ez a paraméter is meg van adva, akkor azt kell megvizsgálni, hogy a fájl létezik-e már:

```
elif [ -f $2 ]; then
       echo "Output file already exists";
       exit 4;
```

Mivel ide akarunk majd írni, ezért azt kell vizsgálni, hogy a fájl nem létezik-e már. Ha létezik, akkor arról a program hibaüzenetben tájékoztatja a felhasználót, majd 4-es hibakóddal befejezi a futást.

Ezután még azt ellenőrzöm, hogy nem adott-e meg a felhasználó több paramétert:

```
elif [ $# -ne 2 ]; then
      echo "Too many parameters"
      exit 5;
```

Ha nem két paraméter van megadva, akkor erről a program tájékoztatja a felhasználót, majd 5-ös hibakóddal leáll a futás.

2.2 A csv-fájl beolvasása

A paraméterek ellenőrzése után, ha azokkal minden rendben volt, akkor megkezdem a csv-fájl beolvasását.

A csv-ben táblázatba olvasható adatokat tárolunk szövegfájl formában, a táblázat sorai a szövegfájlban is sorokként jelennek meg, míg az egyes sorok oszlopait egy meghatározott karakterrel, jelen esetben vesszővel tagoljuk egymástól.

Feladatom során a csv-fájlban a következő adatokat találhatjuk meg soronként:

```
machineName,port,protocol,user,password
```

Azaz:

- **számítógépnév** (vagy IP-cím), amely címen keresztül a gép adatait lekérdezhetjük
- a csatlakozáshoz szükséges **portszám** (a tesztelés során jellemzően 5588 vagy 5589)
- a csatlakozáshoz szükséges **protokol** (jellemzően http vagy https)
- az OpenPegasus elérését lehetővé tevő **felhasználónév**
- a felhasználnévhez tartozó **jelszó**

A fájlt először soronként szeparálva *egy tömbbe olvasom*:

```
CSV=( $(cat $1) );
```

A CSV nevű változó lesz a tömb, amelybe beolvasom a sorokat.

Ezután *végigmegyek az egyes sorokon*, azokat pedig a vesszők mentén feldarabolom. A csv-ben ugyanakkor az első sor fejléc-információt tartalmaz, így azt a sort kihagyom.

```
for((k=1;k<${#CSV[@]};k++));
do
      [..]
done;
```

Beállítom a felbontás karakterét vesszőre (az eredeti tagolókarakter-változót elmentjük):

```
OIFS=$IFS;
IFS=',';
```

Az egy géphez tartozó, a vesszők mentén feldarabolt szövegrészeket az ITEM nevű tömbbe rakom, majd ebből képezek két változót a későbbi elérés meggyorsítására:

```
ITEM=( ${CSV[$k]} );
URL="${ITEM[2]}://${ITEM[3]}:${ITEM[4]}@${ITEM[0]}:${ITEM[1]}";
URLSTART="${ITEM[2]}://${ITEM[3]}:${ITEM[4]}@";
```

Az első segédváltozó (URL) formátuma az alábbi: "protokol://fhnév:jelszó@gépnév:porszám". A második segédváltozó (URLSTART) formátuma pedig a következő: "protokol://fhnév:jelszó@". Ezeket majd csak egyszerűen be kell helyettesítenem, amikor az egyes objektumokat lekérem.

2.3 A gépnév lekérdezése

Miután beolvasom a csv-t és elkezdjük soronként értelmezni azt, minden egyes gépről egyesével lekérdezzük a szükséges adatokat.

Az adatok lekérdezéséhez a CIM-XML linuxos parancssori implementációját, a **wbemcli**-t használom. A wbemcli az alábbi módon működik: meg kell adnom, hogy milyen jellegű adatokra van szükségem (objektumok neveinek felsorolása, objektumok részleteinek felsorolása, adott objektum részleteinek felsorolása stb.), az elérendő gép pontos kapcsolódási paramétereit (protokol://fhnév:jelszó@hoszt:port/), ezután a kiválasztott objektumot, esetleg annak pontos adatait is meg kell adnom.

Először is lekérdezem a géphez tartozó pontos hosztnevet, amelyet a *CIM_OperatingSystem* objektumból olvasok ki (lekérdezés előtt fontos, hogy a darabolókaraker új sorokat daraboljon, tehát vissza kell állítani azt az eredetire, ne pedig a vessző maradjon):

```
HOST=( $(wbemcli ei -nl "${URL}/root/cimv2:CIM_OperatingSystem" | grep -E "^-
CSName=" | grep -v "localhost") );
```

Ezzel a lekérdezéssel több objektum részleteit is visszakaphatnám, ezek közül egyik a localhostként megjelenő gép - ezért ezt kizárom a lekérdezésből a grep -v segítségével. Továbbá több adatot is képes szolgáltatni a lekérdezés a gépről, mi viszont csak a hosztnévre vagyunk kíváncsiak, így csak a "-CSName=" kezdetű sort tartom meg. A lekérdezés pontos sorrendje: végrehajtom a lekérdezést, amelynek egy szöveges kimenete van, ebből kiszűröm csak a hostneveket, majd ebből kizárom a localhost gépet, végül az eredményt elmentem egy változóba.

A HOST változóba ilyen formában kerül a hosztnév:

```
-CSname="hosztnév"
```

Ezért ebből még ki kell nyernem a hosztnevet. Ehhez beállítom a darabolókaraktert idézőjelre ("), felbontom az előbbi kimenetet egy tömbbé az idézőjelek szerint:

```
OIFS=$IFS;
IFS='"';
HNAME=( $HOST );
IFS=$OIFS;
```

Amennyiben a hoszt valamiért nem elérhető (például nem elérhető a hálózaton, vagy nem fut az OpenPegasus szolgáltatás), úgy a wbemcli figyelmezteti a felhasználót a kimeneten, a kiírandó fájlba pedig nem kerül az adott gépről semmi, a szkript futása a következő csv-beli sorra ugrik:

```
if [ -z "$HOST" ]; then
       IFS=$OIFS;
       continue;
```

Ha sikeresen feldaraboltam a HOST változót, akkor elkezdhetem az adatok kiírását a kimeneti fájlba, az alábbi formátumban:

```
++ hosztnév
```

A kiírás az alábbi módon történik:

```
echo "++ ${HNAME[1]}" >> $2;
```

Ezután fognak következni az egyes géphez tartozó hálózati eszközök, majd egy új sor, és a következő gép adatai, míg a csv-ből a bemeneti adatok el nem fogynak.

2.4 A hálózati eszközök lekérdezése

Miután megvan a hosztnév, le kell kérdezni a hálózati eszközöket, és a hozzá tartozó IP-konfigurációs adatokat.

A hálózati eszközök lekérdezéséhez szintén a wbemcli-t használom. A hálózati eszközöket a *CIM_NetworkPort* objektumon keresztül tudom elérni. Először is lekérdezem a beolvasott adatok segítségével az egyes hálózati interfészek adatait biztosító lekérdezési címeket (a gép loopback hálózati eszközét kizárom a megjelenítésből, és így a lekérdezésből is, ez itt történik, a grep -v 'lo"' segítségével) a DEVICES változóba:

```
DEVICES=$(wbemcli ein -nl "${URL}/root/cimv2:CIM_NetworkPort" | grep -v 'lo"'
```

Ez még csak egy felsorolás a hálózati eszközökről, azok pontos lekérdezési címeivel. Ezek alapján fogom lekérdezni a részletes adatokat. Az egyes eszközökön végigiterálva lekérem mindegyik eszköz nevét és fizikai címét:

```
for DEVICE in $DEVICES; do
      LINES=$(wbemcli gi -nl "${URLSTART}${DEVICE}"| grep -E "(^-Name=|^-
PermanentAddress=)");
      [..]
done;
```

Ez a szöveges eredményhalmaz a következő módon jelenik meg:

```
-Name="devicename"
-PermanentAddress="12:34:56:FE:DC:BA"
```

Ezt ismét fel kell darabolnom, ugyanis csak az idézőjelek közti adatokra, a névre és a fizikai címre van szükségem. A feldarabolt értékek közül a fontosakat (név, mac-cím) külön változókba rakom, hogy azokat később könnyebben elérhessem:

```
OIFS=$IFS;
IFS='"';
VALUES=( $LINES );
NAME=${VALUES[1]};
MAC=${VALUES[3]};
```

Miután ezzel megvagyok, a hosztnév alá kiírom az adatokat a kimeneti fájlba, az alábbiak szerint:

```
++ hosztnév
-- eszköznév1, mac-cím
   [..]
-- eszköznév2, mac-cím
   [..]
```

A kiíráshoz végigiterálok a hálózati eszközökön, és megkezdem a kiírást:

```
echo "-- ${NAME[$i]}, ${MAC[$i]}"  >> $2;
```

Minden egyes eszköz kiírása után következik a hozzá tartozó IP-adatok kiírása.

2.5 A hálózati eszközök lekérdezése

A hálózati eszközök lekérése után, azok feldolgozása során lekérdezem az eszközhöz tartozó IP-konfigurációs adatokat. Ezt szintén a wbemcli-vel valósítom meg. Az IP végpontokat a *Linux_IPProtocolEndpoint* objektumon keresztül érem el. Először az előzőekhez hasonlóan csak a végpont pontos címét kérem le, azt egy változóba mentem:

```
IPDEV=$(wbemcli ein -nl "${URL}/root/cimv2:Linux_IPProtocolEndpoint" | grep
"_${NAME[$i]}");
```

A lekérés során csak annak a végpontnak a címét kérem le, amely a specifikus hálózati eszközhöz tartozik (grep-el szűrés a névre), mivel egy hálózati eszközhöz egy IP-végpont tartozik. Ezután a pontos lekérdezési cím segítségével lekérem az objektum összes adatát, és azokból a szükséges adatokra (IP-cím, alhálózati maszk) szűrök:

```
FIELDS=$(wbemcli gi -nl "${URLSTART}${IPDEV}"| grep -E "(^-IPv4Address=|^-
SubnetMask=)");
```

A kapott eredményhalmaz (FIELDS) a következőképp fog kinézni:

```
-IPv4Address="123.123.123.123"
-SubnetMask="255.255.255.0"
```

Ezt a szokásos módon feldarabolom az idézőjelek szerint, egy tömbbe (VALUES) rakom az eredményt, és onnan tudok hivatkozni a szükséges adatokra:

```
IFS='"';
VALUES=( $FIELDS );
IFS=$OIFS;
```

Ha ez megvan, akkor elvégzem az adatok kiíratását:

```
echo "   ${VALUES[1]},${VALUES[3]}"  >> $2;
```

A kimenet végül az alábbi formátumban jelenik meg a szöveges fájlban:

```
++ hosztnév
-- eszköznév1, mac-cím
   IP-cím, alhálózati maszk
-- eszköznév2, mac-cím
   IP-cím, alhálózati maszk
```

Miután a kiírást végrehajtom, visszaállítom a daraboló-karaktert az eredetire, rakok még egy új üres sort a kimeneti fájlba, és a feldolgozás a csv következő sorára ugrik, ha van olyan. Ha nem, akkor a szkript befejezi a futását.

3 Tesztelés

3.1 A tesztkörnyezet felállítása

A szkript teszteléséhez a kiadott *CentOS virtuális gépet* használtam több példányban. Az otthoni routerem segítségével, két fizikai számítógépen összesen 5 virtuális gépet indítottam. Az első fizikai gépen 3 virtuális gép fut, ebből az elsőn futtattam a szkriptet. A három virtuális gépen különböző hálózati beállításokat alkalmaztam:

- **1. gép:** 2 hálózati eszköz

    - 1. eszköz: *bridged*, az itthoni hálózathoz kapcsolódik, a router oszt neki IP-címet
    - 2. eszköz: *host-only*, a VMware VMnet1 hálózatához kapcsolódik
- **2. gép**: 1 hálózati eszköz: *host-only*, VMnet1
- **3. gép**: 3 hálózati eszköz
    - 1. eszköz: *bridged*, az itthoni hálózathoz kapcsolódik, a router oszt neki IP-címet
    - 2. eszköz*: host-only*, a VMware VMnet1 hálózatához kapcsolódik
    - 3. eszköz*: NAT*, a VMware VMnet8 hálózatához kapcsolódik

A *második fizikai gépen két virtuális gép fut*, mindkettő egy-egy hálózati eszközzel, amelyek *bridged* üzemmódban működnek, az itthoni hálózathoz kapcsolódnak.

Néhány gép hosztnevét megváltoztattam, ez látszódik az eredmények listájában. Mivel a hosztnevek alapján a gépek nem elérhetők valamilyen konfiguráció hiánya miatt (pl. pingelni se lehet azokat hostnév alapján), ezért minden gépet (valamelyik) IP-címével azonosítok a bemeneti csv-ben.

A gépeken szükség van az OpenPegasus szolgáltatás futására, ezért minden gépen el kell azt indítanom:

```
sudo /etc/init.d/tog-pegasus start
```

Létre kell még hoznom egy bemeneti csv állományt. Ez a *list.csv* lesz, az alábbi tartalommal:

```
machineName,port,protocol,user,password
192.168.2.12,5989,https,pegasus,LaborImage
192.168.117.129,5988,http,meres,LaborImage
192.168.117.130,5989,https,pegasus,LaborImage
192.168.2.10,5988,http,pegasus,LaborImage
192.168.2.11,5989,https,pegasus,LaborImage
```

Amikor a hibás eseteket tesztelem (főleg a kapcsolódási problémákkal foglalkozókat), akkor a tesztelés gyorsítása miatt kevesebb sorral dolgozom.

Ezután az első virtuális gépre feltöltöm a végleges *getInterfaces.sh* fájlt, futtatási jogot adok a fájlnak, majd ezt futtatom az alábbi tesztesetek szerint.

3.2 Tesztesetek

3.2.1 Sikeres teszt

Bemenet (bash): ./getInterfaces.sh list.csv out.txt

Bemenet(list.csv): *előbb megadott*

Kimenet (bash): *nincs*

Kimenet (out.txt):

```
++ irf2011t1
-- eth1, 00:0C:29:AF:C6:C1
   192.168.2.12,255.255.255.0
-- eth0, 00:0C:29:AF:C6:B7
   192.168.117.128,255.255.255.0

++ irf2011.localdomain
-- eth0, 00:0C:29:F2:2A:CA
   192.168.117.129,255.255.255.0

++ irf2011.localdomain
-- eth0, 00:0C:29:87:43:55
```

```
   192.168.117.130,255.255.255.0
-- eth1, 00:0C:29:87:43:5F
   192.168.2.13,255.255.255.0
-- eth2, 00:0C:29:87:43:4B
   192.168.232.135,255.255.255.0

++ irf2011.localdomain
-- eth0, 00:0C:29:7E:86:30
   192.168.2.10,255.255.255.0

++ irf2011gep2teszt2
-- eth0, 00:0C:29:11:1A:F4
   192.168.2.11,255.255.255.0
```

Mint látható, létező bemeneti fájllal, és nem létező kimeneti fájllal, a specifikációnak megfelelő kimenetet generálja a program.

3.2.2 Sikertelen tesztek

3.2.2.1 Hiányzó bemeneti fájl paraméter

Bemenet (bash): ./getInterfaces.sh

Kimenet (bash): **Missing input file parameter**

Kimeneti fájl *nincs definiálva*

3.2.2.2 Nem létező bemeneti fájl

Bemenet (bash): ./getInterfaces.sh forras.csv result.txt

Bemenet (forras.csv): *nem létező fájl*

Kimenet (bash): **Input file not exists**

Kimenet (result.txt): *nem kerül létrehozásra*

3.2.2.3 Hiányzó kimeneti fájl paraméter

Bemenet (bash): ./getInterfaces.sh list.csv

Bemenet (list.csv): *fent definiált*

Kimenet (bash): **Missing output file parameter**

Kimeneti fájl *nincs definiálva*

3.2.2.4 Már létező kimeneti fájl

Bemenet (bash): ./getInterfaces.sh list.csv out.txt

Bemenet (list.csv): *fent definiált*

Kimenet (bash): **Output file already exists**

Kimenet (out.txt): *a korábban létrehozott out.txt nem fog változni*

3.2.2.5 Túl sok paraméter

Bemenet (bash): ./getInterfaces.sh list.csv out1.txt akarmi

Bemenet (list.csv): *fent definiált*

Kimenet (bash): **Too many parameters**

Kimenet (out1.txt): *nem jön létre*

3.2.2.6 Nem elérhető hálózati eszköz

Bemenet (bash): ./getInterfaces.sh list.csv missingout.txt

Bemenet (list.csv):

```
machineName,port,protocol,user,password
192.168.2.12,5989,https,pegasus,LaborImage
192.168.2.100,5988,http,meres,LaborImage
```

A második IP-cím egy a router DHCP-je által nem kiosztott cím, így az garantáltan nem elérhető.

A program próbálja lekérdezni az adott gépet, azonban nem tudja. A szkript nem timeout-ol ésszerű időn belül (~5 perc), ezért kézzel kell azt lelőni. Ekkor azonban a *missingout.txt*-ben az első gép eszközei még megjelennek:

```
++ irf2011t1
-- eth1, 00:0C:29:AF:C6:C1
   192.168.2.12,255.255.255.0
-- eth0, 00:0C:29:AF:C6:B7
   192.168.117.128,255.255.255.0
```

3.2.2.7 Nem futó OpenPegasus kiszolgáló

Bemenet (bash): ./getInterfaces.sh list.csv nopegasus.txt

Bemenet (list.csv):

```
machineName,port,protocol,user,password
192.168.2.12,5989,https,pegasus,LaborImage
192.168.2.13,5988,http,meres,LaborImage
192.168.2.11,5988,http,meres,LaborImage
```

Kimenet (bash):

```
*
* wbemcli: Http Exception: couldn't connect to server
*
```

Kimenet (nopegasus.txt):

```
++ irf2011t1
-- eth1, 00:0C:29:AF:C6:C1
   192.168.2.12,255.255.255.0
-- eth0, 00:0C:29:AF:C6:B7
   192.168.117.128,255.255.255.0

++ irf2011gep2teszt2
-- eth0, 00:0C:29:11:1A:F4
   192.168.2.11,255.255.255.0
```

Ebben az esetben a csv-ben 2. gépként szereplő gépen leállítottam a Pegasus-kiszolgálót. Mint látható, ilyenkor egy hibaüzenet jelent meg a szöveges kimeneten, a kimeneti fájlba pedig az eszköz nem került bele. Ilyen hiba történik akkor is, ha nem a protokollhoz megfelelő portot adjuk meg a bemenetben.

3.2.2.8 Hibás felhasználónév/jelszó

Bemenet (bash): ./getInterfaces.sh list.csv nouser.txt

Bemenet (list.csv):

```
machineName,port,protocol,user,password
192.168.2.12,5989,https,pegasus,LaborImage
192.168.2.13,5988,http,meresuser,LaborImage
192.168.2.11,5988,http,meres,LaborImage
```

Kimenet (bash):

```
*
* wbemcli: Http Exception: Invalid username/password.
*
```

Kimenet (nouser.txt):

```
++ irf2011t1
-- eth1, 00:0C:29:AF:C6:C1
   192.168.2.12,255.255.255.0
-- eth0, 00:0C:29:AF:C6:B7
   192.168.117.128,255.255.255.0

++ irf2011gep2teszt2
-- eth0, 00:0C:29:11:1A:F4
   192.168.2.11,255.255.255.0
```

Ebben a tesztesetben egy olyan felhasználó nevével próbáltam kapcsolódni a kiszolgálóhoz, amely nem létezik. Ilyen hiba lép fel akkor is, ha a felhasználónak nincs joga olvasni a CIM objektumokat, vagy ha a felhasználónévhez megadott jelszó hibás.

3.2.2.9 Hibás kapcsolódási protokol

Bemenet (bash): ./getInterfaces.sh list.csv noprotocol.txt

Bemenet (list.csv):

```
machineName,port,protocol,user,password
192.168.2.12,5989,https,pegasus,LaborImage
192.168.2.13,5988,tcp,meres,LaborImage
192.168.2.11,5988,http,meres,LaborImage
```

Kimenet (bash):

```
*
* wbemcli: Url Exception: Only supporting http or https protocols
*
```

Kimenet (noprotocol.txt):

```
++ irf2011t1
-- eth1, 00:0C:29:AF:C6:C1
   192.168.2.12,255.255.255.0
-- eth0, 00:0C:29:AF:C6:B7
   192.168.117.128,255.255.255.0

++ irf2011gep2teszt2
-- eth0, 00:0C:29:11:1A:F4
   192.168.2.11,255.255.255.0
```

Ebben a tesztesetben egy a 2. géphez nem http/https protokollal akartam csatlakozni.