Csináld magad – internetes rádióprogramok hallgatása

PCLinuxOS Magazine – 2015. augusztus

Írta: Peter Kelly (critter)

A PCLinuxOS fórumán nemrégiben társalgás folyt az internetes rádiók hallgatásáról. Én nem olvastam ezt a szálat, de Paul Arnote, a magazin szerkesztője figyelte és nagyon érdekelte egy kis szkript, amit dm+ nevú tag készített. Lehetővé tette a felhasználónak, hogy a „Great Little Radio Player” nevű, a PCLinuxOS tárolójában megtalálható kis ingyenes eszköz lenyűgöző állomáslistájából válasszon és játsszon le állomást.

Paul úgy gondolta, hogy ez jó alap lenne egy cikkhez a magazinba. Úgy gondolta, hogy az eljárást egy kicsit tovább kellene fejleszteni egy olyan, a tálcán maradó eszközzé, amivel a felhasználó az internetes rádiókról kedvencek listát készíthet, onnan kiválaszthat egyet lejátszásra, és ha már nem kell, a tálcára lecsukhatja. Paul sajnos eléggé be van táblázva a munka miatt, amit csak tetéz, hogy a fiára is vigyáznia kell, illetve a magazint is el kell készíteni. Mindezek okán nem tudott időt szakítani az ötlet kidolgozására, ezért engem kért meg, hogy foglalkozzam az üggyel. Íme egy nekem szóló e-mailje kivonata:

„A Fórumon a rádióállomások témában most folyó társalgás felkeltette az érdeklődésemet. Amiből az ötletet kaptam, az dr+által használt bash parancs (emlékeim szerint a 3. oldalon olvasható). Zenity-vel választja ki lejátszásra az állomást. Nos, arra gondoltam … mi lenne, ha volna egy olyan bash szkript, ami az asztalra felrak egy „állandó” ablakot, ahol a felhasználó válogathat a „kedvenc” állomások listájából? Plusz, amikor „minimalizálod” az ablakot, az az értesítési részbe csukja le. Amikor ott van, akkor bal egér kattintásra újra megjelenítheti az „állandó” ablakot, vagy állomást lehet váltani a bal kattintás hatására megjelenő kedvencek listájából, a kiválasztott állomások egyikére. Egérrel jobb kattintás hatására választhatsz az „állandó” ablak újbóli megjelenítése és a programból kilépés között. Nem tudom, hogy ez mennyire lehet bonyolult, de úgy vélem, hogy bash szkriptből megvalósítható. Esetleg megoldható úgy is, hogy a kedvencek listázásától a bal kattintásra megjelenő menüben eltekintünk és csak az „állandó” ablak megjelenítése marad. Az újra megjelenített ablakban a felhasználó választhatna akár egy másik állomást is, vagy kiléphet a programból. A stream lejátszását indító parancs az mplayer -playlist [URL] … és jól működik, legalábbis parancssorból.”

Ennek a cikknek az a célja, hogy bemutassa az alap szkriptet, ami néhány olvasó számára hasznos lehet és bemutasson néhány középszintű szkripttelési technikát. Próbálom megmagyarázni a kódot és néhány tulajdonság ötvözésének okát. A kód nem túl összetett, de némi szkriptelési gyakorlat nem árt.

A szkript leírása

Az első dolog, amit készítettem, egy lista az elérendő képességekről.

- Olyan bash szkript legyen, amit az alapbeállítás szerinti PCLinuxOS-ben minden felhasználó futtathat és módosíthat. Szerkesztői megjegyzés: esetleg szükség lehet már telepített GLRP-re (Great Little Radio Player). Ha nem lenne, akkor Synaptic-ból telepíthető.

- A szkript által használt eszközöknek és programoknak már telepítve kell lenniük, ha nem, a PCLinuxOS tárolóiban rendelkezésre állnak.

- A szkript tálca-rezidens alkalmazás legyen, ikonnal és gyorstippel.

- Az egér jobb és bal kattintását érzékelni, és hatására műveletet végre hajtatnikell.

- Ezen műveletek egyike az elérhető funkciók megjelenítése legyen.

- Az alkalmazásból való kilépésre a stream lejátszása álljon meg.

- Az alkalmazás első futtatását észlelje és hozza létre, inicializálja a szükséges fájlokat és könyvtárakat.

- A menü funkciók között legyen kedvenc állomás kiválasztása és új kedvenc hozzáadása a glrp (Great Little Radio Player) által biztosított mesterlistából.

- Az éppen játszott rádió adatfolyam neve megjelenítését valamilyen formába kell önteni.

A fenti lista nagyobb részét szabvány héjprogramozási technikával teljesíteni lehet, de a tálca-rezidens állapot és a egérkattintás érzékelésével menü elérése valami olyasmi, amit korábban még nem próbáltam. Lusta lévén és nem akarván újra feltalálni a kereket, a PCLinuxOS tárolóiban néztem körül olyat keresve, ami meg tud ezek közül valamit csinálni nekem. Találtam egy „alltray” nevű szép kis

eszközt. Átneveztem dm+ közzétett szkriptjét „radio_play”-re és ezt a sort futtattam.

Első próbálkozás

```
alltray ./radio_play -m "Edit List:zenity --text-info --
editable \ --filename=station_list"
```

Noha ígéretes eredményt mutatott, mégsem felelt meg teljesen a célnak, ezért ismét átnéztem az eredeti szkriptet. A szkript jól működik, de nem tálca-rezidens és nem ad fel menüket. A Zenity kiváló rutingyűjtemény, amit sokszor használtam, de 2009-ben leágaztatták és yad néven jelent meg újra (yet another dialog – egy másik dialógus), sok újonnan hozzáadott funkcióval. A yad-ben az értesítő eszköz különösen érdekes, ami ígérete szerint megoldja a tálcaterülethez kapcsolódó problémákat. A yad használata mellett döntöttem.

A szkriptírás elkezdése

Hol kezdjem? Mindig jobban érzem magam, amikor valamit a lemezre írtam és a szkriptjeimnek mindig külön könyvtárat készítek, legalább amikor készen van. Készítettem egy új könyvtárat és beléptem.

```
Mkdir ~/radio_streamer; cd ~/radio_streamer
```

Az alkalmazásnak kell egy ikon, ezért kerestem a /usr/share/icons-ban valami alkalmasat, bemásoltam az új könyvtárba és átneveztem streamer.png-nek.

Ezután megnyitottam a szövegszerkesztőt és készítettem egy radio_streamer.sh nevű fájlt egyetlen sor szöveggel.

```
#!/usr/bin/env bash
```

Ez a sor közli a rendszerrel, hogy a bash héj kell a következő kód értelmezéséhez. A könyvtáram tartalma most így nézett ki:

```
ls
radio_streamer.sh streamer.png
```

Akkor csak meg kell írnom a kódot.

A kódot író emberek problémáinak zöme abból ered, hogy nem követnek egy rendszert a kódban. Én szeretem a változóimat rögtön az indításkor deklarálni és hozzájuk rendelni az esetleg használt karaktersorozatokat. Így a végső kód sokkal könnyebben olvasható lesz.

YAD

A szkript kulcsa a yad értesítő objektuma, ezért elmagyarázom, hogyan használtam ebben a szkriptben. Yados alkalmazás meghívása annak nevével történik, amit kettős kötőjel előz meg. Ezt kettős kötőjellel releváns opciók sorozata követik, amik vezérlik az alkalmazást. Ennek eredménye egy hosszú, esetlen parancs lehet, ezért én a vissza-törtjeles sorkapcsolást alkalmaztam az olvashatóság javítására. A bash-héj ezeket a sorokat egyetlenként kezeli. A yad értesítőjében számos opció alkalmazható, de a szkriptben a sor így néz ki:

```
yad --notification \
--kill-parent \
--listen \
--image="$ICON" \
--text="$HINT" \
--command="bash -c l_click" <&3 &
```

Az első sor indítja ténylegesen a yad értesítőjét. A következő --kill-parent, ami jelet küld a gazda folyamatnak (bash), amikor a yad-értesítő kilép. Az alap jelzés a SIGTERM, így a yad-ot elindító bash folyamat bevégzi, amikor a yad kilép.

A 3. sor az okos --listen közli a yad-dal, hogy a szabványos bemeneten (stdin) várjon parancsot. Még nem határoztunk meg menüt a jobb egérkattintáshoz, amit a --menu=string opcióval lehet megcsinálni, de helyette ezzel az opcióval a menü string az stdin-re is küldhető, ami által menet közben, dinamikusan változtathatjuk a menüt. Az stdin-re küldhető yad-értesítő (notification) parancsok: icon, tooltip, visible, action, menu és quit. Így módunkban áll például a körülményeknek megfelelően cserélni az ikont.

Az ikon beállítása a --image=string opcióval, az elemleírásé pedig a --text=string opcióval történik. Az utolsó sorban határozzuk meg, hogy a tálcaikonra ballal kattintás hatására milyen parancs fusson le, alkalmazzon fájl descriptor 3-at az stdin-re és a háttérben futva adja vissza a vezérlést a bash-szkriptnek.

Fájl descriptorok és csővezetékek használata

A fájl descriptor egy mutató, ami fájlra, vagy adatfolyamra mutat és az elérhető mutatók számozása 0-tól indul. Rendesen 0, 1 és 2 az stdin, stdout és stderr-hez tartozik. Innen tudja a rendszer az írás és olvasás helyét. Mivel nem akarunk más forrásból adatokat küldeni az értesítő objektumunknak, időlegesen át kell írnunk ezeket. A stdin-ünket FD3-ra irányítjuk át, az összes többi folyamat az stdin-jük számára az FD0-t fogja használni,

Mielőtt ezeket az átirányításokat alkalmazhatnánk, be kell ezt állítani, amihez készítünk egy speciális, típusú, csővezetéknek hívott fájlt. A csővezeték

„elsőnek-be-elsőnek-ki” típusú objektum, mint az életben. Ami az egyik végén elsőnek megy be, az elsőnek hagyja el a másik végén. Ezeket „FIFO”-nak is hívják és az ilyet létrehozó Linux-parancs az mkfifo. Ha már van csővezeték fájlunk, használhatjuk a szabványos fájlátirányítást, hogy az FD3-at a csővezetékhez kapcsoljuk.

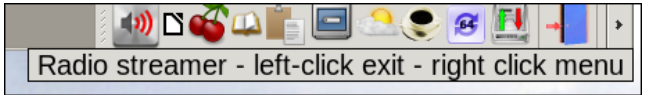
```
mkfifo “$PIPE”
exec 3<> “$PIPE”
```

Változók deklarálása

Ez a témánktól egy kicsit idegen dolog, de menjünk csak egy kicsit vissza. Hivatkoztunk már 'filename'-re, 'string'-re és néhány változóra, ezért ezeket használatuk előtt deklarálni kell, és ahogy azt korábban állítottam, a szkript elején teszem. A bash fejlécsor után, az egyébként üres szkriptfájlhoz add hozzá:

```
HINT=" Radio streamer - left-click exit - right click menu
" (bal klikk kilép, jobb klikk menü)
```

Ez az yad --notification blokkjában fent van meghatározva és akkor jelenik meg,



am
ikor az egérmutató az ikon fölé kerül.

A szkript által használt ikon a szkripttel azonos könyvtárban van és a bash számára a pontos helyének meghatározásához a következő kódsort használhatjuk:

```
$( cd "$( dirname "${BASH_SOURCE[0]}" )" && pwd )/
```

Most akár érted hogyan működik, akár nem, nem kell foglalkozni vele. Mi mind használunk kütyüket a napi életben, mint a telefon, mikrosütő, vagy akármi anélkül hogy tudnánk, valójában hogyan működnek. Nem látok különbséget a programozásban sem. Ez egy kütyü, ami visszavisz abba a könyvtárba, ahonnan a szkriptet futtatjuk, vagyis használd és add az eszköztáradhoz. Történetesen, ha pontról pontra átnézed nem is olyan bonyolult, de észben tartani igen, ezért fájlból másolom és illesztem be mintegy időtakarékosságból.

Az ikon útvonalát és a fájlnevét tartalmazó változó ilyen lesz:

```
ICON=$( cd "$( dirname "${BASH_SOURCE[0]}" )" && pwd
)/streamer.png
```

Add ezt a sort is a szkript-fájlhoz.

A karaktersor, ami a jobb egérkattintásra megjelenő menüt létrehozza a következő formátumot követi

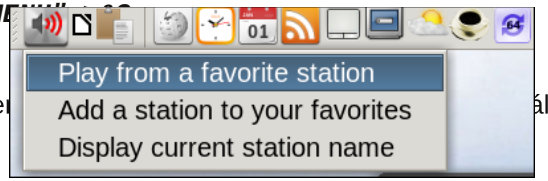
```
title!command|title!command|…
```

A title (cím) egy karaktersor, ami a lenyíló menüben jelenik meg. ! jelzi az adott karaktersor végét. A command a menüelem kiválasztása estén végrehajtandó parancs és a | választja el a menüelemeket. A menüváltozónk definíciója a következő lesz:

```
MENU="Play from a favorite station! bash -c play_favorite|\
Add a station to your favorites! bash -c add_station|\
Display current station name! bash -c show_name"
```

Ezek a furcsa parancsok 'play_favorite', 'add_station' és 'show_name', amik végrehajtására a menüben utasítást adunk, később magyarázom el. A MENU karaktersor az értesítő objektumban megjelenik az stdin-jén (FD3) keresztül.

```
echo "menu:$MENU" >&3
```



Ezt a szkriptben állítása után kell végrehajtani.

A csővezeték fájlnak megkülönböztető nevet kell adnunk és természetesen erre létezik egy mktemp nevű Linux parancs, ami segít ezt megállapítani. Ismét csak nem szükséges pontosan érteni, hogyan működik, csak add ezt a sort a szkripthez.

```
PIPE=$(mktemp -u /tmp/r_streamer.XXXXXXXX)
```

Meg kell mondanunk a szkriptnek, hol találja meg az állomáslistát és erre, ha a Great Little Radio Player telepítve van, ilyen formában találunk rá

```
/home/$USER/.config/glrp/stations.csv
```

Az ehhez hasonló sorok a szkript kód törzsében nem javítják az olvashatóságot, ezért a karaktersort egy változóhoz kapcsoljuk.

```
STATION_LIST="/home/$USER/.config/glrp/stations.csv"
```

Kell még egy hely, ahol tároljuk a saját konfigurációs fájljainkat és a kedvenceink

listáját. A könyvtár legyen:

```
PREFIX="/home/$USER/.config/radio_streamer"
```

A kedvenceknek pedig

```
MY_FAVORITES="$PREFIX/favorites"
```

A függvények – első futás

Amikor a szkript elindul, ellenőriznie kell, hogy első alkalommal hajtják-e végre és a fájlok és azok a könyvtárak, amikben a fájljait keresi ténylegesen léteznek-e és tartalmaznak-e hasznos adatokat. Ha nem, akkor létre kell azokat hozni és némi adattal feltölteni.

Ennek eléréséhez írtam egy függvényt, neve first_run. A függvények egyszerű kódblokkok, amik lefutnak, amikor meghívják. A függvények deklarálásának sorrendje lényegtelen, mivel a kód a memóriába kerül és szükség esetén lefut, ami a függvényeket nagyon hatékonnyá teszi.

A first_run függvényt úgy csináltam, hogy létrehozza a szükséges fájlstruktúrát a szkript többi része számára, így ez lehet az első deklarálandó függvény. Ugyanakkor, a függvény megírása után úgy találtam, legalább egy másik függvény kell neki, hogy működjön. Íme, ahogy én definiáltam a first_run függvényt. A sorszámozás nem része a függvénynek. Csak hivatkozási célból kerültek oda.

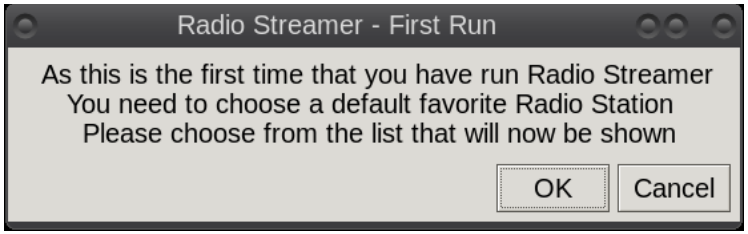
```
1.   first_run() {                # az alap fájlstruktúra
felállítása
2.   if ! [ -d "$PREFIX" ]     # a könyvtár létezik?
3.   then                        # ha nem,
4.      mkdir "$PREFIX"         # készítsd el
5.   fi
6.   if ! [ -s "$MY_FAVORITES" ] # ha létezik és
                                   nem üres
7.   then                        # ha nem
8.      >"$MY_FAVORITES"          # készíts egy üres fájlt
9.   else                        # ha létezik
10.     exit 0                   # Ez nem az első futás
11.  fi
12.                     # a párbeszédben történő dolgok
magyarázata
13.  yad --title "Radio Streamer - First Run" \
14.       --width=550 \
15.       --text=" As this is the first time that you have
6.       run Radio Streamer\n \
17.       You need to choose a default favorite Radio
Station\n \
18.       Please choose from the list that will now be
shown" \
19.       --button="OK:1" \
20.       --button="Cancel:2"
21.       if [[ $? == "1" ]]      # az OK gomb le lett
nyomva
22.   then                    # az add_station függvény
meghívása
23.       add_station      # állomás hozzáadása a
kedvencekhez
24.       killall mplayer  # a más mplayer műveletek
leállítása
                                   és ennek lejátszása
25.       mplayer -nolirc -msglevel all=0 "$s_url" &
26.       echo $s_url > $PREFIX/current_url     # url
mentése
                                                a következő
futáshoz
27.       exit 0     # a first_run függvény elhagyása
28.  else
29.       exit 1     # felhasználói megszakítás
                              szóval vissza a 0-hoz
30.     fi
31.  }                # a first_run függvény vége
```

A megjegyzések elmagyarázzák a függvény működésének nagy részét. A megjegyzések befoglalása a későbbi módosításokat sokkal könnyebbé teszik.

Az 1-5. sor a könyvtár létezését keresi és elkészíti, ha kell.

A 6-11. sor a kedvencek fájlt ellenőrzi. Ha van, akkor nem kell folytatni. Ha nincs, akkor létrehozza és a függvény folytatja tovább.

A 13-20. sor megjeleníti a magyarázó yad párbeszédet. Az OK lenyomására a yad 1-es értéket ad vissza, a Cancel hatására 2-est.



A 21-31. sor vizsgálja a párbeszéd által a $? bash változóval visszaadott értéket, ami mindig az utolsó parancs kilépési kódját tartalmazza. Amikor a first_run függvény visszaadja a vezérlést a fő szkriptnek, a kilépési utasítás által visszaadott érték ellenőrizhető és ha a felhasználó a Cancel gombot nyomta le,

az egész szkriptből ki lehet lépni. Ha a felhasználó az OK gombot nyomta meg, akkor a szkript engedélyezi a folytatást.

```
# is this the first run of the script?
if ! (first_run)
then
      exit # the user cancelled
fi
```

Ez a kód ténylegesen végrehajtja a first_run függvényt és végrehajtja a megfelelő lépést. Minden további függvény végrehajtása a tálca objektummal együttműködésben történik.

Ha az OK-t nyomták le, akkor az add_station parancsot hajtja végre. Mivel nincs ilyen nevű szabvány parancs, ezért nekünk kell elkészítenünk. Ez úgy történik, hogy ezen a néven definiálunk egy függvényt. A kedvenceink listájához állomást adva először az mplayer összes futó műveletét leállítjuk, a kiválasztott streamünket lejátsszuk és mentjük az url-jét a current_url fájlba a következő indításhoz. Ha a Cancel gombot nyomtuk le, akkor a függvény kilép és 1-es értéket ad vissza. Fontos tudni, hogy a visszaadott érték honnan származik.

Az add_station függvény

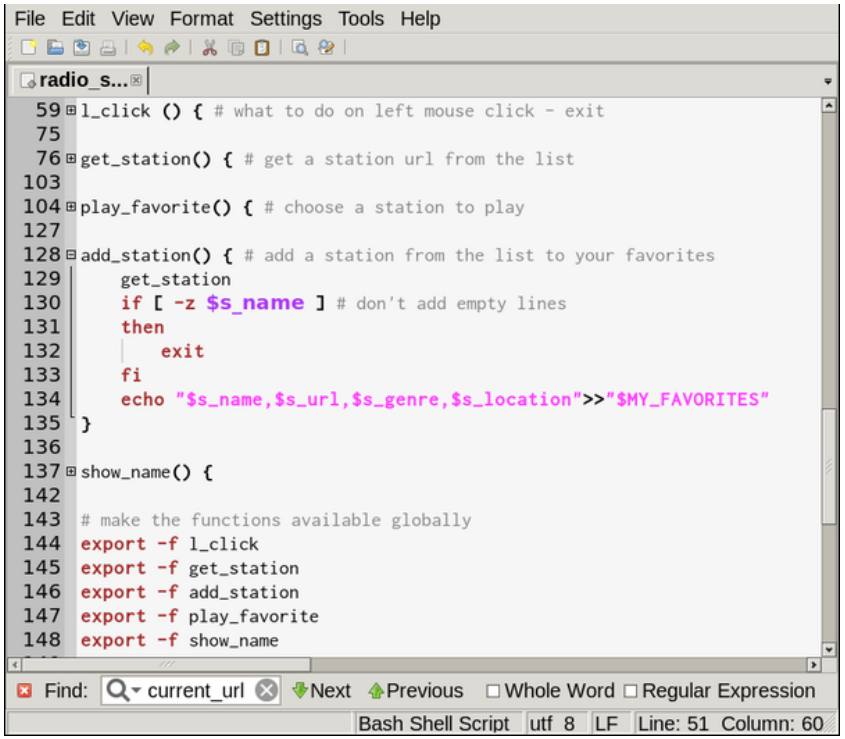
Most meg kell határoznunk az add_station függvényt, ami egyszerű.

```
add_station() { # a listából adj egy állomást a
kedvencekhez
get_station
if [ -z $s_name ] # ne adjál hozzá üres sort
then
      exit
fi
echo "$s_name,$s_url,$s_genre,$s_location">>"$MY_FAVORITES"
}
```

Az add_station függvény egy másik, get_station függvényt hív meg, kilép, ha a felhasználó állomásválasztás nélkül nyomja le az enter-t és néhány változó, vesszővel elválasztott értékét a kedvencek fájl végéhez kapcsolja. Korábban nem láttuk ezeket a változókat, mivel a get_station hozza létre és ahhoz tartoznak, amit még nem definiáltunk. Ha megnézed a MENU string definícióját láthatod, az add_station egyike azoknak a parancsoknak, amik végrehajtását kéri.

Mi jön először?

A first_run függvény hívja meg az add_station-t és az add_station hív egy másik, get_station nevű függvényt. Ez egy tyúk és tojás helyzet arról, hogy melyik függvényt készítsük el előbb és egy olyan függvény meghívása, ami még nem jött létre, zavaró lehet, és némi gondolkodásra van szükség a szkript szerkezetének meghatározásához. Az általam használt módszer, hogy olyan szerkesztőt használok, ami támogatja „kódcsoportosítás”-t (code folding). Sok ilyen szerkesztő van, a hírhedt vim-től a grafikus program-szerkesztőkig, mint a scite és a KDE Kwrite-ja. A kódcsoportosítás lehetővé teszi függvényfejléc írását, amit magyarázó megjegyzések követnek és csak be kell írni a kódot, ha már tudod mit akarsz oda. A függvény törzsébe további megjegyzések írhatók igény szerint és a fejléc kivételével minden elrejthető, ha nem akarod látni. Így sokkal könnyebb áttekinteni a szkript szerkezetét.



A fenti képernyőkép egy Editra nevű szerkesztőt mutat (a PCLinuxOS tárolójában elérhető). A plusz és mínusz gombokra kattintás kibontja, illetve összecsukja a kódot.

Exportálás

Szintén a képernyőképen a 123-126 sorok exportálják a függvényt. Amikor a függvényt meghívják, lefut egy alárendelt héjban, ami nem örökli a szülő héj környezetét. Függvények és változók exportálása teszi lehetővé azok használatát az al-héjakban. A függvényeket -f opcióval kell exportálni. Ezáltal az add_station függvény meghívhatja a get_station függvényt, amit történetesen a szülő héj definiál. Ha a get_station függvény exportálja a s_name, s_url stb. változókat,

akkor elérhetőek lesznek az add_station függvény számára. Exportálnunk kell néhány változót a fő szkriptből, hogy elérhetőek legyenek a függvényeink számára. Add a következőket a változók deklarálása és a függvény kódja közé.

```
# Make some variables available globally
export STATION_LIST
export MY_FAVORITES
export PIPE
```

Vedd észre, hogy az exportált változók pusztán az eredetiek másolatai. Ha az add_station függvény megváltoztatja valamelyik átvett változót, az eredeti változó a get_station-ban változatlan marad.

A get_station függvény

Első látásra komplex függvénynek néz ki, de a kód nagy része pusztán egy yad többoszlopos lista létrehozását szolgálja.

```
1.  get_station()      { # get a station url from the list
2.  chosen=$(cat       "$STATION_LIST" |\
3.  awk    -F, '{print $1"\n"$2"\n"$3"\n"$4}' | \
4.  sed    's/"//g' |\
5.  yad    --list \
6.  --geometry=800x800       \
7.  --title='Internet  Radio Stations' \
8.  --column     Station_Name \
9.  --column     Station_URL \
10. --column     Station_Genre \
11. --column     Station_Location \
12. --hide-column=2    \
13. --no-markup  \
14. --ellipsize=END    \
15. --expand-column=1  \
16. --print-column=0   |\
17. sed    's/|/,/g')
18. s_name=$(echo      "$chosen" | awk -F, '{print $1}')
19. s_url=$(echo       "$chosen" | awk -F, '{print $2}')
20. s_genre=$(echo     "$chosen" | awk -F, '{print $3}')
21. s_location=$(echo  "$chosen" | awk -F, '{print $4}')
22. export       s_name
23. export       s_url
24. export       s_genre
25. export       s_location
26. }
```

A STATION_LIST változó olyan állomáslistára mutat, aminek elemeit kettős idézőjelekkel és vesszőkkel egyaránt elválasztják. Minden sor a következő formátumú

```
“Név”,”URL”,”Műfaj”,”Hely”,”Kedvenc”
```

A yad lista objektum idézőjel nélküli oszlopadatokat vár, amiket új sor karakter választ el és nincsenek idézőjelek. Minden rekordmezőt függőleges vonal „|” karakternek kell elválasztania. A 2-4. sorok hajtják végre az átalakítást.

A 2. sorban az állomások fájl csővezetéken a következő, 3. sorban lévő parancshoz kerül.

A 3. sorban az awk a mezőelválasztó vesszőt állítja be. Az awk ezután kiírja a fájl összes sorának első négy mezőjét, mindegyiket új sor karakterrel választva egymástól és a kimenetet egy függőleges vonallal zárva le. Az ötödik mezőt nem használjuk, ezért egyszerűen elnyomjuk.

A 4. sorban a sed eldobja a dupla idézőjeleket s/”//. A záró „g” globális hatású, azaz a sorban minden előfordulást lecserél. Hatására az összes idézőjel eltűnik.

Most már az adataink a yad által elfogadható formátumúak, átcsövezhetjük a yad-nak, --list, 5. sor.

A 6. sor állítja be a lista párbeszéd dimenzióit.

A 7. sor állítja be a párbeszéd ablak címsorában megjelenített szöveget.

A 8-11. sor az oszlopcímeket határozza meg.

A 12. sor mondja a yad-nak, hogy az URL-ket ne mutassa. Ha kellenek is, megjeleníteni nem kell.

A yad megpróbálja a szöveg megjelenítésénél a „pango jelölő”-t használni. Ez jó, ha színezett, vagy félkövér szöveg kell. Sajnos az importált fájlunk tartalmazhat „&”-hoz hasonló karaktereket, amit jelölőként próbál értelmezni. Ennek elkerülésére kikapcsoljuk a markup-ot a 13. sorban.

Egyes adatok az oszlopmérethez túl hosszúak lehetnek. Az --ellipsize opció lehetővé teszi, hogy a folytatást ellipszisként (…) mutassa, és egyben beállítsa annak pozícióját. A választásom szerint a mező végére teszi a 14. sorban, biztosítván a mező elejének megjelenítését mindenképpen.

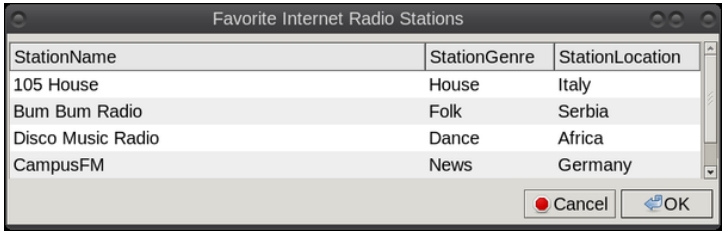
A feltehetően legérdekesebb oszlop az állomások nevei, és a 15. sorban olyan módon lett beállítva, hogy a névből a lehető legtöbbet mutassa meg. Ez egyfajta kompromisszum, mivel az oszlopnév hossza és a párbeszéd által az oszlopszélesség behatárolt. Az oszlop kiterjesztése lehetővé teszi, hogy az összes „fölösleges” szóköz ide kerüljön.

A 16. sor mondja meg a yad-nak, hogy a kiválasztott adatsor mely oszlopa legyen a kimenet. Egy 0 itt azt jelenti, hogy egy oszlopot (az URL-t beleértve) sem kell megjeleníteni. Ezzel a lista párbeszéd beállítása megtörtént, a kimenet átcsövezve a sed parancshoz a 17. sorban. Ez a sed parancs cseréli le a yad által beírt függőleges vonalakat vesszőkre, amikkel a következő awk utasítás könnyebben dolgozik. A 18-21. sorok mindegyike a kimeneti mezőket változókhoz rendeli a 23-26. sor exportálja azokat.

A play_favorite függvény

Ez a függvény a get_station-hoz nagyon hasonlít.

```
1. play_favorite() { # choose a station to play
2. current=$(cat "$MY_FAVORITES" | \
3.     awk '!x[$0]++' | \
4.     awk -F, '{print $1"\n"$2"\n"$3"\n"$4}' | \
5.     yad --list \
6.     --geometry=800x800 \
7.     --title='Favorite Internet Radio Stations' \
8.     --column Station_Name \
9.     --column Station_URL \
10.    --column Station_Genre \
11.    --column Station_Location \
12.    --hide-column=2 \
13.    --no-markup \
14.    --ellipsize=END \
15.    --expand-column=1 \
16.    --print-column=2 |\
17.    sed 's/|$//'
18. )
19. echo $current > $PREFIX/current_url # save it for next
run
20. killall mplayer
21. mplayer -nolirc -msglevel all=0 "$current" &
22. show_name
24. }
```



Ez a jobb kattintás menüből meghívható másik függvény. A kedvencek fájl az, ami most a lista párbeszédhez lesz átcsövezve. A cím megváltozik és az mplayer-t a kiválasztott URL-lel hívja meg. A két mplayer-nek átadott opció a távvezérlő hozzáférését szünteti meg és kikapcsol minden szövegkimenetet, mivel ezek egyikére sincs szükségünk.

Az awk parancs a 3. sorban érdekes. Nem magyarázom el, de ez másik kis „idő megtakarításom”. Eltávolítja a duplikált sorokat. A kedvencek fájlban sok ismétlés fordulhat elő. Ez önmagában nem probléma, de nem akarjuk a megkettőzött sorokat megjeleníteni itt. Ha meg akarod tisztítani a fájlt, akkor egy ilyen parancsot használj (következő oldal)!

```
uniq -u <(cat favorites) > new_favorites ; mv -f
new_favorites favorites
```

Ez a sor folyamat helyettesítést alkalmaz a duplikált sorok kiszűrésére az eredetiből és a szűrt adatok kiírására egy új fájlba. A régi fájt ezután az új felülírja. Előbb készíts másolatot!

A 17. sor eltávolítja a záró függőleges karaktereket a yad által adott kimenetből. A 19. sor menti az újonnan kiválasztott url-t a current_url fájlba a további használatra. A 20. és 21. sor állítja elő az új kimenetet és a 22. sor hívja meg a show_name függvényt, amit még nem írtam meg.

A show_name függvény

Ezt a függvényt a play_station függvény hívja meg a jobb kattintásos menün keresztül, a szkript működésének legelején, amikor az utoljára játszott állomást a current_url fájlból lekérdezi.

Elhatároztam, hogy ehhez a függvényhez egy kis értesítő eszközt fogok használni, neve notify-send, ami pár másodpercre feldob egy kis ablakot a képernyő sarkában és eltűnik az útból.

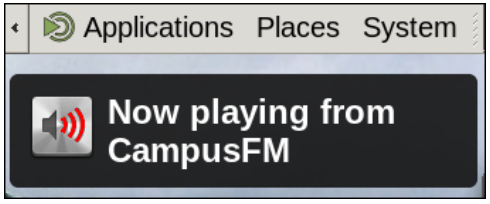
A notify-send a következő opciókat használja a függvényhez:

```
-t    időszak milliszekundumban, amíg látható marad.
-i    ikonban a megjelenítendő fájlnév.
```

És a megjelenítendő szöveg.

```
show_name() {
current_name=$(grep $(cat $PREFIX/current_url)
$PREFIX/favorites | \
awk -F, '{print $1}')
notify-send -t 3000 -i $ICON "Now playing from
$current_name"
}
```

Az aktuálisan játszott URL állomásnevének kinyeréséhez a grep parancsot használjuk, hogy keresse meg a kedvencek fájlban, majd az awk-val emelje ki az állomásnevet és tegye a current_name változóba. A notify-send parancs a $ICON-t használja az ikon megjelenítéséhez, 3 máasodperc erejéig (3000 ms) lesz látható, és megjelenít némi előre szerkesztett szöveget és a current_play változóban lévő értéket.

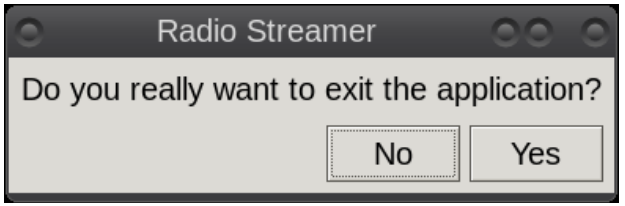


A l_click függvény

Az objektum ikonján ballal kattintva lehetővé válik a kilépés és az mplayer bezárása. Ezt túl könnyű véletlenszerűen elérni, ezért megerősítés kell, hogy a felhasználó valóban ezt akarja.

```
l_click () { # mi történja a bal kattintásra - exit
      exec 3<> "$PIPE"  #az átirányítás elérhetővé tétele
                                    a függvényen belül
      yad --title "Radio Streamer" \
          --text=" Do you really want to exit the
application? " \
          --button="No:1" \
          --button="Yes:2"

            if [[ $? = "1" ]]
            then
  exit
            else
     echo "quit" >&3    # a yad-nak a kilpés parancs
küldése
     killall mplayer
     rm -f "$PIPE"          # a fifo ürítése
            fi
}
```



A függvényre az értesítő beállító blokkjában van hivatkozás a --command opcióval. Amikor meghívják, a függvény megjeleníti a jóváhagyó párbeszédet, és ha a felhasználó NO-t nyom, akkor a függvény kilép, a szkript fut tovább. Ha a YES-t választja, akkor a „quit” parancsot küldi a yad-nak az FD3-on keresztül, a mplayer folyamat befejeződik és a csővezeték fájl törlődik. A szkript befejeződik.

A szkript szerkezet

Csak ismétlésképpen, a szkript szerkezete ilyen kell legyen:

bash fejléc
változók deklarálása
változók exportálása
függvénydefiníciók
függvények exportálása

a first_run függvény meghívása
a csővezeték fájl elkészítése
adatátirányítás a csővezeték mechanizmuson
a tálca mechanizmus beállítása és háttérbe rakása
a tálca objektum számára a menü karaktersor megjelenítése

Ha olyan szerkesztőt használsz, ami felajánlja a kódcsoportosítást, egy kis görgetéssel azt a szerkezete fogod látni.

Működési hibák (bugs)!

Tapasztalatom szerint nincs tisztességes program hiba nélkül. Az első, amit ebben a szkriptben észrevettem, hogy az állomásváltáskor a korábbi kimenet az új mellett tovább működött. Ezt azzal oldottam meg, hogy ezt a sort adtam hozzá

```
killall mplayer
```

az új stream lejátszását indító sor elé a play_favorite függvényben.

A következő probléma az volt, hogy néhány állomást nem játszott le, még ha ugyanezen állomást parancssorból közvetlenül meghívva lejátszotta. Ezt az okozta, hogy a yad függőleges vonal karaktert hozzáadva továbbította az url-t az mplayer-nek. Egy egyszerű sed utasítás kiszűrte ezt. A play_favorite függvény most így néz ki.

```
      --ellipsize=END \
      --expand-column=1 \
      --print-column=2 |\
      sed 's/|$//'
)
killall mplayer
mplayer -nolirc -msglevel all=0 "$current" &
}
```

Kétségkívül további hibák is jelentkezni fognak és kijavításuk további, előre nem látható problémákat okoz. Ez a szép a programozásban.

Tulajdonságok hozzáadása

Az mplayer-t filmek vetítésére készítették, de nagy képességű hangok stream-elésére is, ahogy itt is csináltuk. Sokkal több, a kézkönyvlapokban dokumentált tulajdonsággal is bír, amit esetleg használhatsz. Például, az mplayer képes a hangadatfolyam vételére és fájlba írására. Ez hasznos lehet, ha olyasmit hallgatnál meg, amit nem megfelelő időben sugároznak. Az adatfolyam mentésére egy stream.mp3 fájlba ilyen sort használj:

```
mplayer -dumpaudio -dumpfile stream.mp3 stream_url
```

Valós URL-t rakj a stream_url helyére. Indítható és leállítható olyan eszközzel mint a cron, vagy az at. Digitális művek jogosulatlan tárolásának jogi következményei lehetnek a lakóhelyeden.

A teljes szkript

Alábbiakban látható a teljes szkript. Letöltheted még a magazin weblapjáról, innen.

A szerkesztő megjegyzése: nem nehéz a szkriptet sokkal „több tulajdonsággal” felruházni. Lehetővé teheted új állomás kézi bevitelét is ahelyett, hogy kizárólag a GLRP-vel csomagoltakra támaszkodnál. Emellett lehetővé teheted a „Kedvenc állomások” listájának szerkesztését is (a mostani állapot szerint szövegszerkesztővel meg kell nyitnod a ~/.config/radio_streamer/favorites listát és törölni a továbbiakban szükségtelenek a kedvencek listájából). Ezekhez és további fejlesztésekhez a tanulás érdekében teret hagyunk, fel kell venned a kesztyűt!

```
#!/usr/bin/env bash

# Initialise the strings
MENU="Play from a favorite station! bash -c play_favorite|\
Add a station to your favorites! bash -c add_station|\
Display current station name! bash -c show_name"
HINT=" Radio streamer - left-click exit - right click menu
"
PIPE=$(mktemp -u /tmp/r_streamer.XXXXXXXX)
ICON=$( cd "$( dirname "${BASH_SOURCE[0]}" )" && pwd
)/streamer.png
PREFIX="/home/$USER/.config/radio_streamer"
STATION_LIST="/home/$USER/.config/glrp/stations.csv"
MY_FAVORITES="$PREFIX/favorites"

# Make some variables available globally
export STATION_LIST
export MY_FAVORITES
export PIPE
export PREFIX
export ICON
```

```
# define functions
first_run() { # set up basic file structure
# check for and create if necessary
# directory ~/.config/radio-streamer
if ! [ -d "$PREFIX" ]
then
    mkdir "$PREFIX"    # create the directory
fi

# check for and create if necessary
# the file favorites
if ! [ -s "$MY_FAVORITES" ]
then
    >"$MY_FAVORITES"    # create the empty file
else
    exit 0 # This is not the first run
fi

yad --title "Radio Streamer - First Run" \
    --width=550 \
    --text="      As this is the first time that you have
run Radio Streamer\n  \
      You need to choose a default favorite Radio Station\n
\
      Please choose from the list that will now be shown" \
       --button="OK:1" \
       --button="Cancel:2"
if [[ $? == "1" ]]
then
    add_station # add a station to the favorites file
    killall mplayer # play it
      mplayer -nolirc -msglevel all=0 "$s_url" &
      echo $s_url > $PREFIX/current_url # save it for next
run
#     show_name
    exit 0
else
    exit 1 # user decided to cancel
fi
}

l_click () { # what to do on left mouse click - exit
    exec 3<> "$PIPE"    # make the redirection available
within the function
    yad --title "Radio Streamer" \
        --text=" Do you really want to exit the application?
" \
        --button="No:1" \
        --button="Yes:2"

    if [[ $? = "1" ]]
    then
        exit
    else
        echo "quit" >&3         # send yad the quit command
        killall mplayer
        rm -f "$PIPE"     # remove the fifo
    fi
}

get_station() { # get a station url from the list
chosen=$(cat "$STATION_LIST" |\
    awk -F, '{print $1"\n"$2"\n"$3"\n"$4}' | \
    sed 's/"//g' |\
    yad --list \
    --geometry=800x800 \
    --title='Internet Radio Stations' \
    --column Station_Name \
    --column Station_URL \
    --column Station_Genre \
    --column Station_Location \
    --hide-column=2 \
    --no-markup \
    --ellipsize=END \
    --expand-column=1 \
    --print-column=0 |\
    sed 's/|/,/g')
s_name=$(echo "$chosen" | awk -F, '{print $1}')
s_url=$(echo "$chosen" | awk -F, '{print $2}')
s_genre=$(echo "$chosen" | awk -F, '{print $3}')
s_location=$(echo "$chosen" | awk -F, '{print $4}')

export s_name
export s_url
export s_genre
export s_location
}

play_favorite() { # choose a station to play
current=$(cat "$MY_FAVORITES" | \
    awk '!x[$0]++' | \
    awk -F, '{print $1"\n"$2"\n"$3"\n"$4}' | \
    yad --list \
    --geometry=800x800 \
    --title='Favorite Internet Radio Stations' \
    --column Station_Name \
    --column Station_URL \
    --column Station_Genre \
    --column Station_Location \
    --hide-column=2 \
    --no-markup \
    --ellipsize=END \
    --expand-column=1 \
    --print-column=2 |\
```

```
sed 's/|$//'
)
echo $current > $PREFIX/current_url # save it for next run
killall mplayer
mplayer -nolirc -msglevel all=0 "$current" &
show_name
}

add_station() { # add a station from the list to your
favorites
    get_station
      if [ -z $s_name ] # don't add empty lines
      then
      exit
      fi
    echo
"$s_name,$s_url,$s_genre,$s_location">>"$MY_FAVORITES"
}

show_name() {
current_name=$(grep $(cat $PREFIX/current_url)
$PREFIX/favorites | \
      awk -F, '{print $1}')
notify-send -t 3000 -i $ICON "Now playing from
$current_name"
}

# make the functions available globally
export -f l_click
export -f get_station
export -f add_station
export -f play_favorite
export -f show_name

# is this the first run of the script?
if ! (first_run)
then
    exit # the user cancelled
fi

# set up the pipe mechanism
mkfifo "$PIPE"
exec 3<> "$PIPE"

# set up the tray object
yad --notification \
    --kill-parent \
    --listen \
      --image="$ICON" \
      --text="$HINT" \
      --command="bash -c l_click" <&3 &

# send the menu string to the tray object
echo "menu:$MENU" >&3
mplayer -nolirc -msglevel all=0 $(cat $PREFIX/current_url)
&
show_name
```