Két új PDF-szkript (GUI-val) az eszköztáradba

PCLinuxOS Magazine - 2026. május

Írta: Paul Arnote (parnote)

Elképzelheted, PDF fájlokkal foglalkozom... mintegy ... ÁLLANDÓAN. Így, a számítógépemre szinte minden parancssori eszközt telepítettem, amiről úgy gondolom, hogy hasznos lehet a PDF-fájlokkal való munkához. De még könnyen elfeledkezhetek róluk, és olyan eszközhöz nyúlhatok, ami grafikus felhasználói felülettel rendelkezik.

Miért, kérdezhetnéd? Nos, elsősorban azért, mert ha parancssori eszközt használnék, úgy kotorászok magam körül, mint egy csirke, aki a magvakat kaparja össze, miközben próbálok emlékezni a parancssori eszközök opcióira, hogy megtaláljam, amit kell. Egy idő után beleunok, amitől még csábítóbbá lesz egy grafikus felhasználói felületű (GUI) eszköz választása.

A GUI, amihez ilyenkor nyúlok, a Master PDF Editor. A Master PDF Editor leginkább egy szövegszerkesztőre hasonlít, aminek PDF az alapértelmezett kimenete. Ráadásul a PDF-fájlok többségénél lehetővé teszi a képek vagy szövegek cseréjét is, majd ezeknek a változtatásoknak a mentését PDF-fájlba. Félreértés ne essék... ez egy hatalmas, erőteljes program, ami kiválóan teszi a dolgát. De néha túlzás használni, néha pedig túl lassú.

Nem úgy lassú, hogy a program lassan futna. Nézzünk egy lehetséges esetet. Tegyük fel, hogy hozzá kell férnem a képekhez egy PDF fájlban. Mint, ahogy PDF maga is az. Betölthetem a Master PDF Editorba, és egyenként elmenthetem a fájl képeit. Ez könnyedén lehet több mint 30 perc. Végül megvannak a szükséges képek, de marad bennem az érzés, hogy kell legyen gyorsabb, kevésbé munkás mód a képeknek a kimentésére.

Szerencsére van.

A PCLinuxOS Magazine 2026. áprilisi számának Tip Top Tips rovatában leközöltük kalwisti egy, tippjét. Kalwisti közzétett a tippjeiből egy „kivonatot”-t a fórumon, és egy hivatkozást egy PDF-dokumentumra, a tipp teljes leírásával. Tehát a 2026. áprilisi számban történő újranyomtatáshoz hozzá kellett férnünk a PDF szövegéhez és a képeihez (a formátuma nem felelt meg a Magazin elrendezésének).

Így hát betöltöttem a PDF-et a Master PDF Editorba, és egyenként rákattintottam és elmentettem a képet... 16 mentés. Ez körülbelül 15–20 percet vett igénybe. És ekkor kezdtem el azon gondolkodni, hogy „kell legyen gyorsabb, könnyebb módja” a képek kimentésére!

Ekkor eszembe jutottak az általam felhalmozott parancssori PDF-eszközök (mind a PCLinuxOS adattárból lett telepítve). Amikor átnéztem az eszközöket és opcióikat, rájöttem, hogy megérettek egy saját szkript létrehozására. Természetesen, gondoskodtam arról, hogy az egyéni szkript GUI elemekkel rendelkezzen, a Zenity jóvoltából.

Biztos írhattam volna kalwistinek, és ugyanannyira biztos, hogy nagyon rövid időn belül elküldte volna nekem. De miért zavarnám kalwistit, ha nem muszáj? Ez indított el a bash-szkript készítési kirándulásomra. Miután megvolt a szkript előzetes működő verziója, készítettem egy másodikat is, a PDF-fájlok összes szövegelemének kimentésére.

„Könnyű mint egy pite”

Az első szkript, amit csináltam, kimenti az összes képet a PDF-fájlból. **PIE**-nak (pite) hívom, a **PDF Image Extractor** rövidítéseként.

Mint sok általam írt egyéni bash-szkript, ez is futtatható önállóan parancssorból, vagy használható Thunar egyéni műveletként (ha nem tudnád, HATALMAS Xfce rajongó vagyok). Beírhatod a szkriptet kézzel is, vagy letöltheted a magazin szerveréről. Ha megvan, mentsd a fájl(oka)t abba a könyvtárba, ahol az összes szkriptedet tárolod. Ezután távolítsd el a “.txt” fájlkiterjesztést, és tedd végrehajthatóvá a szkriptet.

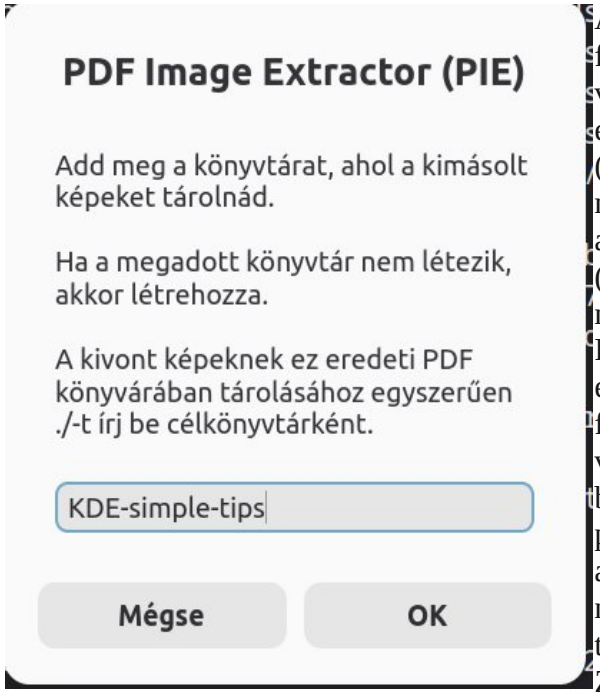
Letöltheted a PIE-t innen, ha akarod. A szkript teljes fájlmérete 2,0 KiB, így a szó szoros értelmében egy szempillantás alatt le kell töltődnie.

Remélhetőleg a könyvtár, ahol a szkripteket tárolod, a $PATH utasításban megtalálható. Így csak a szkript nevét kell megadni, majd az esetleges parancssori paramétereket. Ha a könyvtár, ahol a szkripteket tárolod, NEM szerepel a $PATH-ban, akkor a szkript teljes elérési útját, a nevét és a szükséges parancssori paramétereket kell használni. Amint látod, a szkripteket olyan könyvtárban kell tárolni, ami a $PATH utasítás része, az SOKKAL egyszerűbbé teheti az életed.

Mielőtt a szkript boncolgatásába kezdenénk, vessünk egy pillantást a teljes szkript.

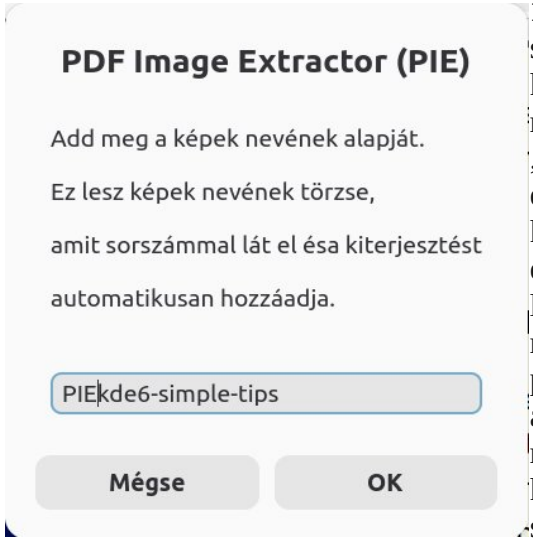
```
#!/bin/bash

# Ez a bash szkript a GPL v. 2.0 licenc alatt ingyenes, és
# szabadon terjeszthető az említett licenc feltételeinek megfelelően.
# Írta: Paul Arnote, a The PCLinuxOS Magazine főszerkesztője
# a The PCLinuxOS Magazine 2026. májusi számához.

#Használat: PIE.sh <PDF fájl neve>
# Thunar egyéni műveletként is használható, egyéni konfigurációval
# a parancssora PIE.sh %n. Fájlminta: *.pdf;*.PDF
# Megjelenés: „Egyéb fájlok“
# Ez a szkript az összes érvényes képet kivonja egy PDF-fájlból
# és menti azokat egy kiválasztott alkönyvtárba. Ha az adott könyvtár
# nem létezik, létrehozza.

INPUT=${1,,}
DFLT=`basename $INPUT .pdf`
DIR=$(zenity --entry --width=300 --height=250 --title="PDF Image Extractor (PIE)" --text="Add meg a könyvtárat, ahol a kimásolt \nképeket tárolnád.\n\nHa a megadott könyvtár nem létezik,\n létrehozza.\n\nA kivont képeknek ez eredeti PDF-\nkönyvárában tárolásához egyszerűen\n./-t írj be célkönyvtárként.\n" --entry-text=$DFLT)
if [ $? == 1 ]; then
   exit
fi

DIR=$DIR
if [ ! -d ./$DIR ]; then
        mkdir ./$DIR
fi
sleep 1

BASE=$(zenity --entry --width=350 --title="PDF Image Extractor (PIE)" --text="Add meg a képek nevének alapját.\n\nEz lesz képek nevének törzse,\n amit sorszámmal lát el ésa kiterjesztést\n automatikusan hozzáadja.\n\n" --entry-text="PIE-"$DFLT)
if [ $? == 1 ]; then
   exit
fi

pdfimages -all $1 ./$DIR"/"$BASE | zenity --progress --title "PDF Image Extractor (PIE)" --width=350 --text="Kérem várj ...\n\nKivonás folyamatban...\n" --pulsate --auto-close --auto-kill

cd ./$DIR

if [ $DIR == "./" ]; then
FILE=$(pwd)
else
FILE=$DIR
fi

COUNT=$(ls -1f | grep $BASE | wc -l)
zenity --info --width=250 --title="PDF Image Extractor (PIE)" --text=$COUNT" képfájlt vont ki\n a PDF-fájlból és\nmentett ide\n\n $FILE."

exit 0
```

Mint minden bash szkriptnél, az első sor az ú.n. „shabang” indít el mindent. A **3-15. sorok** csupán megjegyzések, amik leírják (röviden) a szkript célját, a kiadási feltételeket (GPL v2.0) és néhány rövid használati megjegyzést.

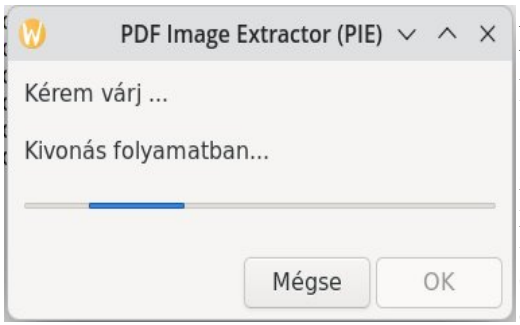


A **16. sorban**, a bemeneti fájl teljes fájlneve kisbetűre változik, majd beállítunk egy „globális” változót (DFLT) (**17. sor**), hogy megtartsa a PDF fájl nevét, amiből képeket nyerünk ki (alapértelmezés szerint marad a PDF neve, de a PDF fájlkiterjesztést eltávolítjuk, és a teljes fájlnevet kisbetűre változtatjuk). A **18. sor** beállítja az első Zenity párbeszédpanelt. Ezen belül adunk a végfelhasználónak néhány rövid utasítást a teendőkről, beállítjuk a Zenity párbeszédpanel szélességét és magasságát, és beállítunk egy alapértelmezett alkönyvtárat a kivont képek tárolására (--entry-text=$DFLT). Természetesen a felhasználó felülírhatja, ha beír egy másik könyvtárnevet. Ha tudott, hogy a PDF-fájl nem tartalmaz sok képet, nem okozhat túl nagy gondot ugyanabban a könyvtárban tárolni, amiben a PDF is van. De ha egy PDF-ben sok a kép (a tesztelés során a The PCLinuxOS Magazine 2026. márciusi számából gyűjtöttem ki a képeket... mind a 262-t; Meemaw egy másik PDF magazinból vett ki képeket, és végül több mint 1400 kép lett), sokkal rendezettebb marad, ha a képeket annak a könyvtárnak az alkönyvtárába bontja ki, ami azt a PDF-fájlt tartalmazza, amelyből képeket szeretnéd kinyerni. Hé... akár megpróbálhatnánk itt rendet és rendezettséget tartani!

Természetesen, nem vagy ***köteles*** az alapértelmezett (már beírt) értéket elfogadni. Bármilyen könyvtárnevet szabadon használhatsz. Ne feledd, hogy bármi is legyen annak a könyvtárnak neve, az a képek kinyerésére szánt PDF-fájlt tartalmazó alkönyvtáraként lesz beállítva.



19-21. sorok teszik lehetővé a szkriptnek, hogy „elegánsan” kilépjen, ha a felhasználó megnyomja a párbeszédablak „Cancel“gombját. A **23-27. sorok** ellenőrzik, hogy létezik-e a könyvtár, amibe a képeket kell elhelyezni, és ha nem, akkor létrehozza azt. A **29. sor** meghívja a második Zenity párbeszédpanelünket, és megkéri a végfelhasználót, hogy adjon meg egy „törzs-” nevet a kibontott képfájlokhoz. A szövegbejegyzés „defaults” (alap) értéke PIE-<default-base-name>. Ez azt jelenti, hogy az összes kép PIE-<alapnév>-vel kezdődik, a fájlnév végén sorszámmal. Ennek a képelnevezési konvenciónak használatával az összes képet csoportosítva tudjuk tartani. És csakúgy, mint a könyvtár párbeszédpaneljénél, szabadon megadható a kívánt alapnév. A „default” érték elfogadása véleményem szerint csak megkönnyíti a dolgokat.



Ahogy korábban, a **32-34 sorok** teszik lehetővé a szkript „elegáns” kilépését, ha a felhasználó a Zenity párbeszédpanel „Cancel“gombjára kattintana. A képek PDF-fájlból való kinyerésére a „komoly munka” a **36. sorban** történik. A **pdfimages** az a parancssori eszköz, ami a képek kinyeréséért felelős. Az -all paraméter mondja meg a pdfimages-nek, hogy bontsa ki az összes felismert formátumú képet. A **./**

$DIR”/”$BASE paraméterrel mondjuk meg a pdfimages-nek, hogy milyen könyvtárba, alkönyvtárba mentse a képeket. Ezután ezt a parancsot a Zenity folyamatjelző párbeszédpanelére továbbítjuk egy pulzáló folyamatjelzővel, így tudjuk, hogy a szkript még mindig működik.

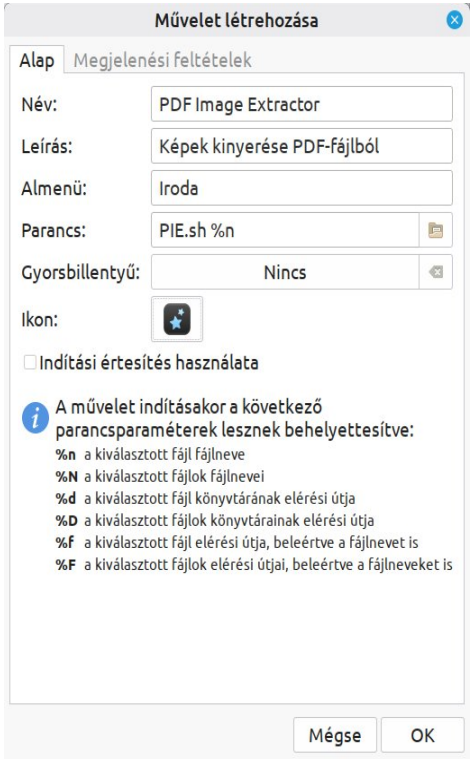
A **38. sorban** kiadjuk a **cd** parancsot, hogy a szkriptet biztosan a jelenlegi könyvtárunkban tartsuk. A **40.-től 44.-ig sorok** állítják be a kibontott képek mentési helyének megjelenítéséhez a formátumot egy olyan párbeszédben, amit én „confirmation”-nak (nyugtázás) hívok, ami jelzi, hogy a feladat lefutott. A **46. sorban** megszámoljuk, hogy hány fájlt vontunk ki a PDF-ből, így meg tudjuk jeleníteni azt a nyugtázó párbeszédpanelen.

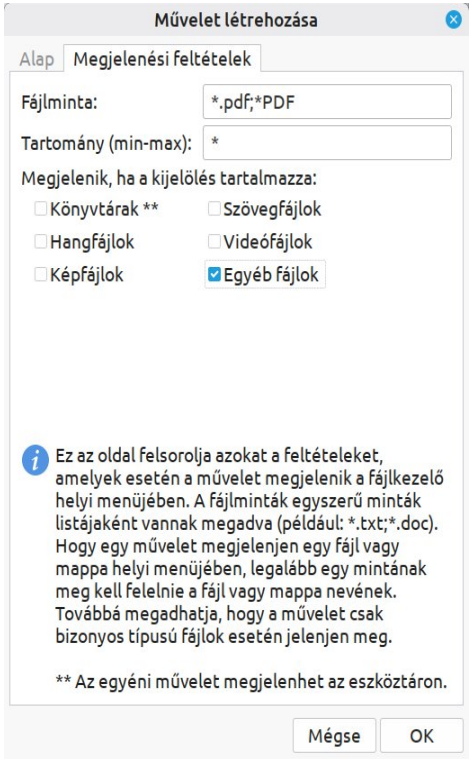
A **47. sor** beállít egy „Information” Zenity párbeszédpanelt, ami a feladat befejezésekor jelenik meg. Megmondja, hány képfájlt bontottak ki, és hol tároltuk azokat. Az **exit 0** parancs a **49. sorban** lehetővé teszi a szkript elegáns kilépését.

PIE futtatása

Ahogy korábban említettem, a szkript futtatható önállóan parancssorból, vagy Thunar egyéni műveletként. Nincs kétségem afelől, hogy más fájlkezelőknél is, mint például a Dolphin és a Caja, hozzáadható a szkript a felhasználói menühöz, de ez túlmutat e cikk keretein (és túlmutat a jelenlegi képességeimen, mivel én mindig csak Xfce.t futtatok).

A terminálból önálló parancsként való futtatásához a fájl csak egy argumentumot igényel. Annak a PDF-fájlnak a nevét, amiből képeket szeretnéd kinyerni. Amikor a szkriptet terminálból futtatod, segít a megfelelő helyre navigálásban, hogy a cd paranccsal belép a abba a könyvtárba, ami tartalmazza azt a PDF-fájlt, amiből képeket szeretnéd kinyerni, majd onnan fut. Ezután már csak a Zenity párbeszédpanelek utasításait kell követni a feladat elvégzéséhez szükséges adatok bevitelével.

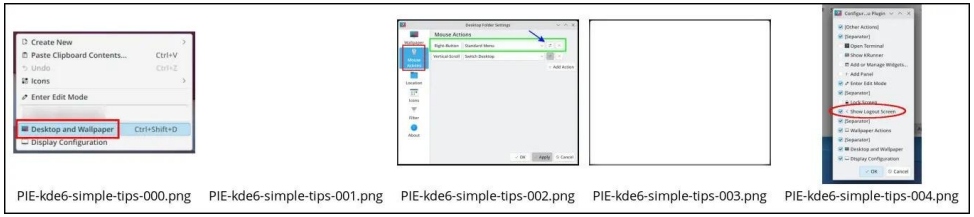




Hogy a szkriptet Thunar egyéni műveletként beállítsd, az egyéni műveletek párbeszédpaneledet alakítsd a fent láthatónak megfelelővé. A bal oldali képen (a „**Alap**” lapot mutatja) töltsd ki a „Név” mezőt (PDF Image Extractor), a „Leírás” mezőt (Képek kinyerése PDF-fájlból), az „Almenü” mezőt (ha használsz ... nálam az „Iroda” almenüben található) és a „Parancs” mezőt (PIE.sh %n).

A „**Megjelenési feltételek**” lapon állítsd be a „Fájlmintát” (*.pdf;*.PDF), hagyd a „Tartomány” mezőt az alapértelmezett „*” értéken, és helyezz egy jelölést az „Egyéb fájlok” előtti négyzetbe. Miután ezeket mind kitöltötted, mentsd az „OK” gombra kattintva.

Ne feledd, hogy a szkriptjeim könyvtára (/home/parnote/bin) a $PATH utasításomban megtalálható, ami azt jelenti, hogy közvetlenül meghívhatom a anélkül, hogy elérési utat kellene megadnom hozzá. Ismétlem, ha a szkripteket olyan könyvtárban tárolod, ami NEM SZEREPEL a $PATH-ban,



akkor a „Parancs” mezőben meg kell adni a szkript teljes elérési útját.
Fent, kalwisti PDF-jéből kivont pár kép látható, ahogy a Thunar-ban megjelennek. Néhány kép „üres”, a képek PDF-ben tárolási módja miatt. Nem vagyok a PDF-ek belső szerkezetének „szakértője”, de az első kép „a“ kép, míg az „üres“ kép az alfa csatorna adatait tartalmazza, így a PDF fájl képes átlátszó PNG fájlok megjelenítésére. Az üres fájlokat megtarthatod, vagy törölheted. Ez tőled függ.

A **pdfimages** eszköz számos grafikus fájlformátumot ismer. Némelyikük elég könnyen azonosítható... JPEG, PNG, TIFF, JPEG2000 (JP2), JBIG2, CCITT. Felsorolja, ha beírod a „pdfimages --help”-t egy parancssorba. A „többi” grafikus formátum támogatása nem olyan könnyen azonosítható, mivel nincsenek a felsorolásban. Például itt, a The PCLinuxOS Magazine-ban már néhány éve WEB-képeket használunk. Annak ellenére, hogy a WEBP grafikák nem szerepelnek a pdfimages repertoárjában, tudja „kezelni”. A PIE szkript „tesztelésem” során azt láttam, hogy a pdfimages „a” képet JPG fájlként bontja ki, majd ugyanabból kibont egy PNG-t is, hogy az átlátszósághoz alfa csatornaként szolgáljon. A Magazin készítésekor használt képek közül soknak nincs átlátszósága, de a pdfimages-nek fogalma sincs arról, hogy van-e, vagy sem. Szóval, még ha egy képnek nincs is átlátszósága, a pdfimages továbbra is kivonja „a” képet JPG fájlként, és ugyanazt a képet PNG-ként, hogy alfa-csatorna képként szolgáljon az áttetszőség biztosítása érdekében.

Elmondhatom, hogy pdfimages a AVIF-képeket „nem hivatalosan” támogatja. A „szokásos” PDF-készítő eszközeim egyike sem képes AVIF-fájlok használatára, így a Scribus sem, amivel sok PDF-et készítek, az eszköz, amivel évek óta havonta elkészítjük a PCLinuxOS Magazine-t. Tehát az AVIF fájlkompatibilitás „tesztelésére” egy képet AVIF formátumba konvertáltam (az ImageMagick segítségével), majd kinyomtattam egy PDF fájlba a GIMP-ből. Amikor kivontam a képet a PDF fájlból, a pdfimages tökéletesen érvényes PNG fájlként írta ki a képet a meghajtómra.
Mindazonáltal nagyon elégedett leszel a PIE működésével. A számítógépemen „megmértem” a szkriptet úgy, hogy a parancssorból futtattam, „time” parancsot követően. Kalwisti PDF-jénél mindössze 8,558 másodperc alatt 30 képet bontott ki (valójában csak 16 kép van, de ne felejtsd, hogy az imént tárgyalt okok miatt további képek készültek). Ebbe a teljes időbe beletartozik az a 3,291 másodperc, amíg várta, hogy megnyomjam a „Enter” billentyűt (csak elfogadtam az összes alapértelmezést), és 0,139 másodpercet a rendszerre.

A PIE tökéletes? Nem. A tesztelés során Meemaw és én találtunk pár PDF-et, amiből a képek nem lettek jók. De a PIE képes volt kinyerni a képeket az általunk megadott PDF-fájlok túlnyomó többségéből. Nehéz pontosan kitalálni, hogy azoknak a PDF-fájloknak a készítésénél mit csinált másként. Azt is nehéz kitalálni, hogy a készítést irányítók vagy projektmenedzserek mire gondoltak. Anélkül, hogy tudnánk *pontosan*, mit tettek vagy mire gondoltak, a legtöbb „következtetés”, amire juthatunk, puszta spekuláció. Nem szabványos vagy nem támogatott képformátumot használtak? Egyéni színpalettát tartalmaznak? Hacsak nem voltunk jelen a PDF létrehozásakor, nincs mód arra, hogy pontosan megtudjuk, mit és miért tettek.

Abban azonban egészen biztos vagyok, hogy rá fogsz jönni, ahogy Meemaw és én is a szkript tesztelése során, hogy a PIE képes kezelni az esetlegesen rádobott PDF-fájlok túlnyomó többségét, és sikeresen kinyerni a képeket belőlük.

Új PET a házban

Az általam készített második szkript a PIE-munkámból született. Gyakran nem elég csak a képeket kinyerni a PDF-fájlból. Bizonyos esetekben a szöveges elemeket is ki kell nyerni és menteni egyszerű szövegfájlként.

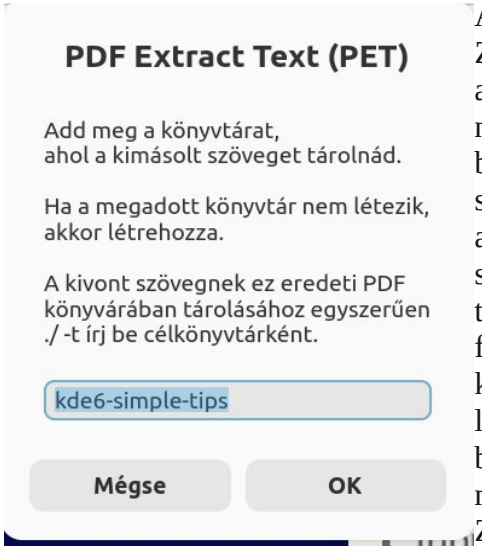
Hogy hozzájuss a PET-hez, a PIE-nél leírtak mérvadók. Innen letöltheted szkriptet. A PET-szkript fájl 1,8 KiB méretű, vagyis szó szerint egy szempillantás alatt le kell töltődnie.

Tehát, mielőtt belefognánk a szkript boncolgatásába, vessünk rá egy pillantást.

```
1.  #!/bin/bash
2.
3.   # Ez a bash szkript a GPL v. 2.0 licenc alatt ingyenes, és
4.   # szabadon terjeszthető az említett licenc feltételeinek megfelelően.
5.   # Írta: Paul Arnote, a The PCLinuxOS Magazine főszerkesztője
6.   # a The PCLinuxOS Magazine 2026. májusi számához.
7.
8.   #Használat: PET.sh <PDF fájl neve>
9.   # Thunar egyéni műveletként is használható, egyéni konfigurációval
10.  # a parancssora PET.sh %n. Fájlminta: *.pdf;*.PDF
11.  # Megjelenés: „Egyéb fájlok“
12.
13.  # Ez a szkript az összes szöveget kivonja egy PDF-fájlból
14.  # és menti egy kiválasztott alkönyvtárba. Ha az adott könyvtár
15.  # nem létezik, létrehozza.
16.
17.  INPUT=$(1,,)
18.  DFLT=`basename $INPUT .pdf`
19.
20. DIR=$(zenity --entry --width=300 --height=250 --title="PDF Extract Text
(PET)" --text="Add meg a könyvtárat,\nahol a kimásolt szöveget tárolnád.\n\nHa a
megadott könyvtár nem létezik,\nakkor létrehozza.\n\nA kivont szövegnek ez eredeti
PDF\nkönyvárában tárolásához egyszerűen\n"./" -t írj be célkönyvtárként.\n" --
entry-text=$DFLT)
21.    if [ $? == 1 ]; then
22.       exit
23.    fi
24.  DIR=$DIR
25.  if [ ! -d ./$DIR ]; then
26.             mkdir ./$DIR
27.  fi
28.  sleep 1
29.
30. BASE=$(zenity --entry --width=350 --title="PDF Extract Text (PET)" --
text="Adj egy fájlnevet a kivont\nszövegnek.\n\nA .txt fájlkiterjesztést
automatikusan\nhozzáadja és az itt megadott néven\n
menti el a fájlt..\n\n" --entry-text=$DFLT)
31.    if [ $? == 1 ]; then
32.       exit
33.    fi
34.
35.  pdftotext $1 ./$DIR"/"$BASE".txt" | zenity --progress --title "PDF Extract
Text (PET)" --width=350 --text="Kérlek várj...\n\nKivonás folyamatban...\n" --
pulsate --auto-close --auto-kill
36.
37.  cd ./$DIR
38.
39.  if [ $DIR == "./" ]; then
40.            FILE=$BASE".txt"
41.  else
42.            FILE=$DIR"/"$BASE".txt"
43.  fi
44.
45.  zenity --info --width=250 --title="PDF Extract Text (PET)" --text="A szöveget
a(z) "$1",\nfájlból kimentette ide \n\n./"$FILE
46.
47.  exit 0
```

A PET-szkript első 16 sora gyakorlatilag megegyezik a PIE-szkript első 16 sorával, a “images”-re hivatkozások “text.”-re változtatásával. Szóval, nem látom szükségét, hogy újra átmenjünk ezeken a sorokon.

A **17. sor** módosítja a bemeneti fájl nevét kisbetűsre, hogy biztosítsa az törzsnév parancs megfelelő működését a **18. sorban**. Az törzsnév parancs leválasztja a PDF fájl kiterjesztését a bemeneti fájlról, hogy beállítsa az alapértelmezett értékeket a Zenity párbeszédpanelein.

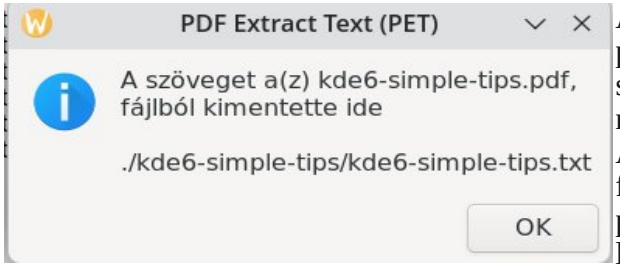


A **20. sorban** beállítottuk az első Zenity párbeszédpanelt. Ebben az ablakban adunk a felhasználónak néhány rövid utasítást a teendőkről, beállítjuk a Zenity párbeszédpanel szélességét és magasságát, és egy alapértelmezett alkönyvtárat a kivont szövegek tárolására (--entry-text=$DFLT). Természetesen a felhasználó felülírhatja, ha másik könyvtárnevet ír be. A **21-23. sorok** lehetővé teszik a szkript tiszta befejezését, ha a végfelhasználó megnyomja a "Cancel" gombot a Zenity párbeszédpanelen.

A **24-28. sorok** ellenőrizik, hogy létezik-e a megadott könyvtár. Ha nem, akkor a megadott könyvtárat elkészíti.

A **30. sorral** a második Zenity párbeszédpanel létrejön, ami lehetővé teszi a felhasználó számára, hogy megadja a szövegkivonat törzsnevét. Elfogadhatja az alapértéket, de megadhat másikat. A **31-33. sor** biztosítja a szkript a tiszta, korrekt befejezésének mód-ját, ha a végfelhasználó a Zenity párbeszédpanel „Cancel" gombját kiválasztja.

A „komoly munka" a **35. sorban** történik, ahol a **pdftotext** eszköz kivonja a szövegelemeket a PDF-ből. Ez átmegy egy Zenity folyamatjelző párbeszédpanelre folyamatjelzővel. Hogy őszinte legyek, kétlem, hogy valaha is látni fogod ezt. A szövegkinyerés olyan gyors, hogy csak egyszer-kétszer láttam, és akkor is csak egy másodpercre vagy rövidebb ideig. Ennek ellenére szükségesnek éreztem, hogy beleírjam, hátha valamikor belefutsz egy igazán nagy PDF fájlba, aminél egy kicsit tovább tart a szöveges elemek kinyerése. Így tudod, hogy a szkript fut, ahelyett, hogy csak bámulnád a semmit, nem tudván, hogy történik-e valami.



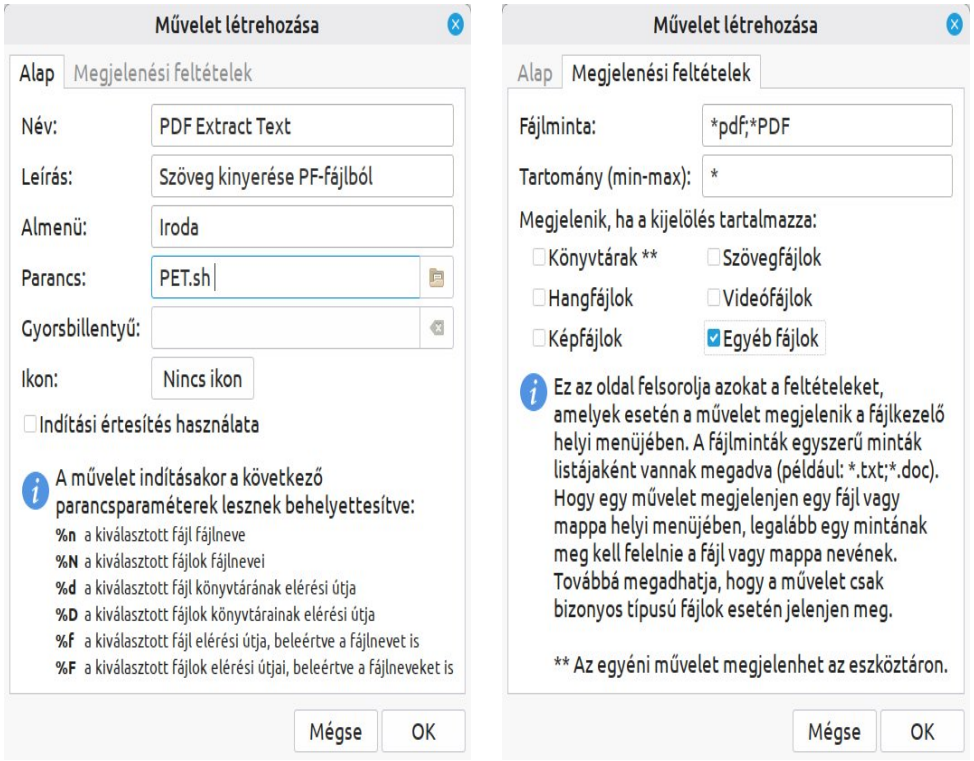
A **37. sorban** kiadjuk a **cd** parancsot, csak hogy a szkript biztosan a tudja a megfelelő munkkönyvtárat. A **39-43. sorok** állítja be és formálja meg a megerősítő párbeszédpanelen megjelenő szöveget a feladat befejezésekor. Majd a **45. sor** megjeleníti a feladat befejeződését a jelző Zenity párbeszédpanelt, amit én szeretek a megerősítő párbeszédnek nevezni. Tájékoztatja a végfelhasználót, hogy a szöveges elemeket kivonta a PDF-ből, és hol tárolta az eredményül kapott ".txt" fájlt.

És végül a **47. sor** teszi lehetővé a szkript a tiszta kilépését a feladat befejezését követően.

Futó PET

A PIE-hez hasonlóan a PET is csak egy parancssori opciót használ, ez pedig annak a PDF-fájlnak a fájlneve, amelyből ki kellene bontani a szövegelemeket. És csakúgy, mint a PIE esetében, a PET futtatható önálló szkriptként a parancssorból, vagy Thunar egyéni műveletként is.

Ha terminálból futtatod, legjobb, ha cd-vel belépsz PDF-fájlt tartalmazó könyvtárba, amiből ki szeretnéd kibontani a szövegelemeket. Segít, hogy a dolgok SOKKAL inkább rendben legyenek, és megóv a teljes elérési utak megterhelő használatától az utasítás parancssori paraméterként.



Hogy a szkriptet Thunar egyéni műveletként beállítsd, az egyéni műveletek párbeszédpaneledet tedd hasonlóvá a fent láthatóhoz. A bal oldali képen (a „**Alap**” lapot mutatja), töltsd ki a „Név” mezőt (PDF Extract Text), a „Leírás” mezőt (Szöveg kinyerése PDF-fájlból), az „Almenü” mezőt (ha használsz ... nálam az „Iroda” almenüben található) és a „Parancs” mezőt (PET.sh %n).

A „**Megjelenési feltételek**” lapon állítsd be a „Fájlmintát” (*.pdf;*.PDF), hagyd a „Tartomány” mezőt az alapértelmezett „*” értékre állítva és helyezz egy jelölést az „Egyéb fájlok” előtti négyzetbe. Miután ezeket mind kitöltötted, mentsd az „OK” gombra kattintva.

Ne feledd, hogy a szkriptjeim könyvtára (/home/parnote/bin) a $PATH-ban megtalálható, így közvetlenül meghívhatom a szkriptet anélkül, hogy elérési utat kellene megadnom hozzá. Ismétlem, ha a szkripteket olyan könyvtárban tárolod, ami NEM SZEREPEL a $PATH utasításban, akkor a „Parancs” mezőben meg kell adni a szkript teljes elérési útját.

```
 1 Financial security doesn't just happen. It takes
 2 planning and commitment and, yes, money.
 3
 4 FACT
 5
 6 Today, only 43 percent of Americans
 7 have calculated how much they need to
 8 save for retirement.
 9
10 FACT
11
12 In 2005, of those who had 401(k)
13 coverage available, 25 percent didn't
14 participate.
15
16 FACT
17
18 The average American spends 20 years
19 in retirement.
20
21 To find out more, call the Employee Benefits
22 Security Administration at 1-866-444-EBSA (3272)
23 and request the following brochures:
24 Savings Fitness:
25 A Guide to Your Money and Your Financial Future
26 Taking the Mystery Out of Retirement Planning
27 What You Should Know about Your Retirement Plan
28 Filing a Claim for Your Retirement Benefits
29 Choosing a Retirement Solution for Your Small Business
30 Women and Retirement Savings
31 Or view them on the Web at: www.dol.gov/ebsa
32
33 The following Web sites can also be helpful:
```

A fenti képen egy PDF-fájlból kinyert mintaszöveg látható, amit Meemaw osztott meg velem a szkriptek tesztelése során (a szövegfájl Mouspad-ben van megnyitva, a sorszámozás be van kapcsolva). Amint látod, ez tényleg csak egyszerű szöveg. A szöveg “újrafelhasználásához” (feltéve, hogy rendelkezel a szöveg felhasználásához szükséges jogokkal ... ne felejts el számolni a szerzői jogok megsértésének következményeivel) némi formázásra lesz szükség, függetlenül attól, hogy milyen eszközt használsz a reprodukálásához. De legalább megvan a szöveg, tehát az újraformázás csak valami, amit meg kell tenni, amikor szerkeszted.

Említettem, hogy a PET is *G Y O R S*?! Még a PIE-nél is gyorsabban végzi a dolgát.

Összegzés

Tehát két hatékony eszközt adhatsz hozzá az eszköztárhoz, amiket, ha használsz, SOK időt takaríthatsz meg. Ezek az eszközök gyorsak abban, amit csinálnak, és követik a régi Linux-gyakorlatot, „csinálj egy dolgot, és csináld jól.”

Többféle forgatókönyvet tudok elképzelni a felhasználásukra, ahol ezek a szkriptek hasznosak lehetnek. Ráadásul jelentős időt takarítanak meg. Tegyük fel, hogy felül kell vizsgálnia egy három éve létrehozott dokumentumot. Frissíteni szeretnéd, vagy kell a dokumentumot. Bár nem találod az eredetit, de sikerül megtalálni a dokumentum PDF-fájlját. Ezekkel az eszközökkel kinyerheted a képeket és a szöveget a PDF-ből, majd újra elkészítheti... természetesen a változtatásokkal.

Már vettem hasznát ezeknek a szkripteknek PDF-fájlok készítésében, valamint a képek és szövegek kinyerésében. Valójában több olyan eset is eszembe jut, amikor bárcsak ez benne lett volna az eszköztáramban. Időnként nekünk (akik itt a Magazinban vagyunk) szükségünk van PDF-fájlban lévő képek elérésére. Korábban PDF-megjelenítőben kellett megnyitnunk, és képernyőképet kellett készítenünk a képről... és remélni a legjobbakat. Ezek az eszközök sokkal könnyebbé tették volna.