A Z-Shell

Írta: Peter Kelly (critter)

Mi ez?

Amikor beviszel egy parancsot a Linux-parancssorban, akkor szöveget írsz be a héjba, ami közted és az operációs rendszer között fordítóként működik. Ez általában áttekinthető és a felhasználó egyszerűen begépeli a parancsát, vagy végrehajtja a szkriptjét anélkül, hogy tudna ilyen evilági dolgokról.

A legtöbb Linux disztribúció által alapból használt héj az általában bash-ként említett Bourne Again SHell. A Unix felhasználók az erőteljesebb Korn shell-t, ksh-t, vagy csh-ként, vagy c-shell-ként ismert c nyelvi shell-t használhatják. Ha sok időt töltesz a parancssorral, akkor érdemes lehet megpróbálnod az ennél is erősebb zsh-ként is ismert z-shell-t, ami a PCLinuxOS tárolójában elérhető.

Mire képes?

Íme néhány azok közül az okok közül, amik miatt a zsh-n érdemes gondolkodni:

• parancskiegészítés, ami kiterjed mind a parancsokra, mind az opciókra és argumentumokra;
• helyesírás-ellenőrzés;
• kötegelt átnevezés;
• témázhatóság és kiterjesztett prompt, beleértve a jobbra rendezést és az automatikus elrejtést;
• kiterjesztett „dzsóker”-használat;
• többszörös átirányítás tee nélkül;
• betölthető modulok olyanok kiterjesztésére, mint matematikai rutinok és hálózati együttműködés;
• többsoros parancsszerkesztés;
• sokkal teljesebb kiterjesztés és helyettesítés;
• menüválasztás;
• változók szerkesztése.

Sokkal több van, de ez ad némi képet a zsh képességeiről.

Ezt hogyan csinálja?

A GNU eszközöknek, szövegalapúak lévén, szükségük van egy beviteli és szövegsor-szerkesztési módszerre. Ezt a lehetőséget a readline könyvtár biztosítja. A bash shell és sok „beépített” funkciója ezt a könyvtárat használja, hogy a néha elég lenyűgöző szerkesztési tulajdonságokat, mint a parancskiegészítés, biztosítsa.

A z shell ezt fejlesztette tovább a saját z-shell sorszerkesztőjét, zle-t kialakítva. Ez a szerkesztő általában visszafelé kompatibilis a bash szerkesztési tulajdonságokkal, de sokkal több hasznos „időspórolóval” és trükkel rendelkezik. Elérhetők opciók, amik a zsh-t jobban, vagy kevésbé teszik kompatibilissá a bash-sel és sokkal kötöttebb shell-lekkel, mint az sh, ash és dash, vagy a POSIX shell (amit zavaróan és konzekvensen sh-nak neveznek).

A z-shell leginkább a ksh-hoz, korn shell és legkevésbé a csh-hoz, c-shell hasonlít. A bash-ban járatos felhasználóknak leginkább a sok további eszköz tűnik fel, amiket szívesen bekapcsolnának. Minden más azonnal használható, vagyis nem kell újratanulni a parancssori szintaxist. Ez alól egy kivétel van (noha nem volt vele problémám), a többszavas változók meghatározása, de még ennek is van olyan opciója, ami lehetővé teszi a sokkal bash-szerűbb működést, ha ez fontos lenne számodra.

Sokkal több az eltérés a zsh-ban a szkriptelés és a tömbkezelés tekintetében. Például a tömbelemek indexelése 0-tól indul a bash-ban és egytől a zsh-ban. Ez a tulajdonság megváltoztatható a KSH_ARRAY opcióval, de az összes létező bash szkripted esetén nem lesz gond, köszönhetően #!/bin/bash-nak az első sorban, ami közli az operációs rendszerrel, hogy bash-t használjon (vagy bármi mást, amit ide raktál /bin/sh /usr/bin/perl...) a fájl parancsainak értelmezésére.

Az említett képességek nem mindegyike van alapból bekapcsolva, és az első futás eléggé unalmas lehet (az unalmasság a parancssorban nagyon jó dolog). Láthatod az első futtatás indító képernyőjét. Itt lehetőséged van aktiválni néhányat azok közül a tulajdonságok közül, amik a zsh-t megkülönböztetik a sokkal hagyományosabb héjaktól.

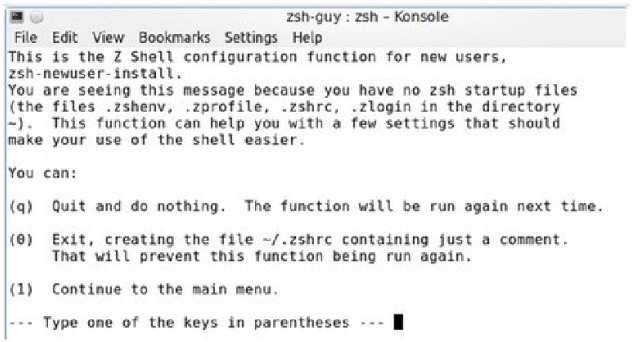
Az olyan disztribúciók wikijében, mint az Arch és a Gentoo, részletes dokumentáció van a zsh-ról, de kifejezetten jelzik, hogy a zsh saját disztribúcióra fordított/csomagolt változatára érvényes. Ebben a bevezetésben bemutatnám, hogyan állítsuk fel és futtassuk a PCLinuxOS által adottat, hogyan állítsuk be és használjunk néhányat a hasznos tulajdonságai közül és talán felébresszem az étvágyad, hogy beruházz a jövőbe.

Telepítés és kezdeti beállítás

Nyisd meg a Synaptic-ot, frissíts és jelöld ki az osszes frissítést, majd jelöld be a zsh-t és a zsh-doc-ot telepítésre, vagy parancssorból, végül is ez a cikk arról sor.

```
# apt-get update
# apt-get dist-upgrade
# apt-get install zsh zsh-doc
```

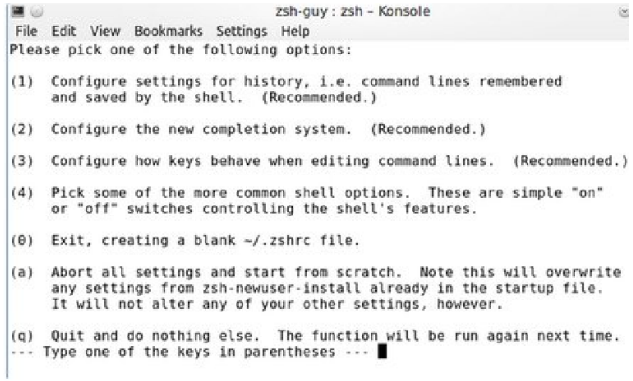
Az első két parancs opcionális, de ajánlott.



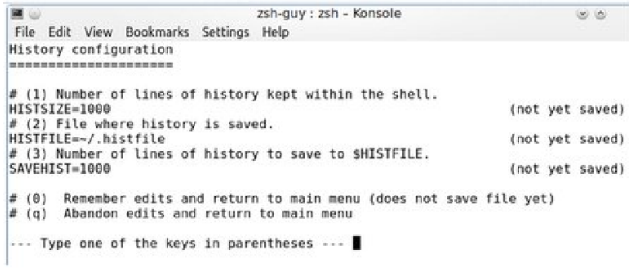
Nyiss parancssort (még mindig bármilyen, általad általában alkalmazott shell-t használhatsz, feltehetően bash-t) és írd be $ zsh.

Ez megnyit egy z-shell példányt és mivel nincs semmilyen beállított konfigurációs fájl, a következő indító képernyő jelenik meg.

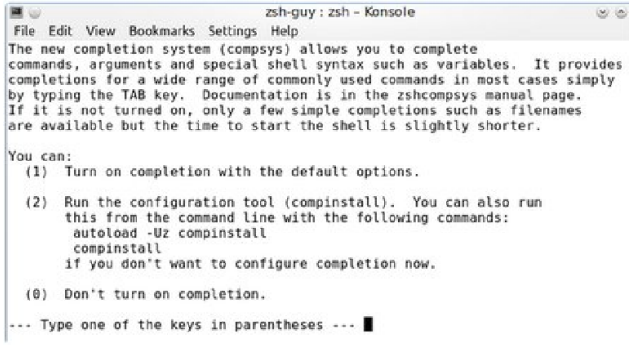
Ez és a következő képernyő lehetővé teszi, hogy beállítsd a kezdeti munkakörnyezetet.



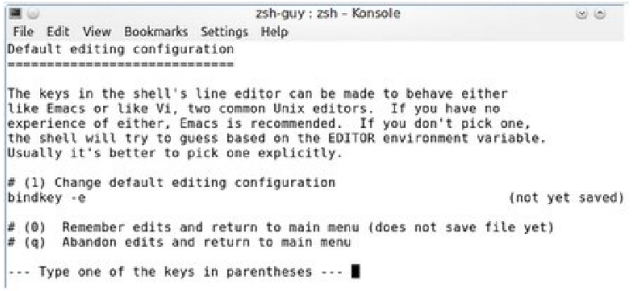
1-et lenyomva a lent bemutatott menühöz hasonló menüt kapsz. Ezeket az opciókat a helyi zsh konfigurációs fájlba menti, így később egyszerű szövegszerkesztővel átírhatod.



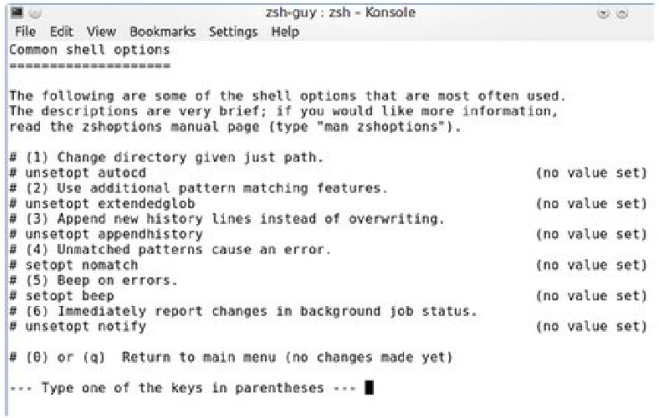
Az 1-es opció az alábbi képernyőt hozza fel. Ha csak nincsen különleges okod ezeknek a beállításoknak a megváltoztatására, akkor írj be 0-t. Semmit sem ment az eszközből kilépésig, amíg jóvá nem hagyod.



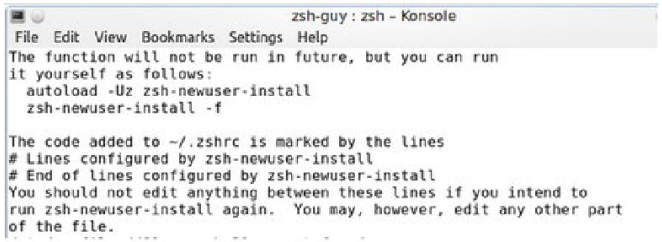
A 2-es opcióval a következő képernyőt kapod. Most csak 1-est írj az alapbeállításokhoz. A zsh-ban a kiegészítés nagyon összetett és az opciók elég zavarosak lehetnek.



3-as opció ezt a képernyőt adja. Ha a vi editort használod, késztetést érezhetsz a szerkesztő gombok ahhoz történő igazítására, de a bash emacs billentyűket használ alapból (pl. CTRL+a a sor elejére ugráshoz). Így, ha hozzászoktál ezekhez a gyorsbillentyűkhöz, jobb ha amellett maradsz a zavar elkerüléséhez.



A 4-es opció egy kicsit érdekesebb. A shell opciót 1-re, vagy 2-re javaslom állítani. Írd be a opció számát majd válaszd a set-et (beállít).



Végül térj vissza a főmenühöz és válaszd a beállításaid mentése opciót. Most a konfigurációs fájlt kiírja, ideje a zsh-t alap héjnak kijelölni. Ezt a /etc/passwd fájl root-ként történő szerkesztésével, vagy a következő paranccsal lehet:

```
chsh -s /bin/zsh
```

Kérni fogja a jelszavadat, majd ki kell jelentkezni a változtatások érvényre juttatásához.

Néhány terminálemulátor, mint a kde konsole és a mate-terminal profilt használ, amelynél meghatározható, hogy mely héj lépjen be indításakor, ezért a profilt szerkeszteni kell a zsh-hoz.

Olvashatod a figyelmeztetést

```
/etc/profile.d/01msec.sh:21: = not
found.
```

A figyelmeztetést az msecs egyik biztonsági szkriptje adja, ami ellenőrzi, hogy az aktuális könyvtár a PATH-ban megtalálható-e (ez egy ismert biztonsági ügy és a PCLinuxOS okosan alapból „no”-ra állítja).

Az üzenetet az okozza ahogy a zsh az ellenőrzéseket kezeli. A kód amiatt panaszkodik, ami a /etc/profile.d/01msec.sh 21. sorában található:

```
if [ "$ALLOW_CURDIR_IN_PATH" == "yes"
]; then
```

A probléma a „==” kezelésével van. A régi stílusú operátorokat [ ] cseréld az előnybe részesített [[ ]] operátorra, a zsh elégedett lesz, eltűnik a hibaüzenet. Ez teljesen biztonságos, mivel a teszt eredménye ugyanaz. Végül is nekem nem volt gondom. Ahogy tudom, az új stílus a POSIX szabványban nincs benne, ezért tudd, ha a megfelelés gondot okozna.

Tudd, bármely, a .bashrc-ben, vagy a profilban meghatározott alias és függvény itt elérhetetlen lesz, mivel ez új környezet (noha a függvények a bash-ból exportálhatók az alárendelt shell-be export -f-fel).

Parancskiegészítés

Most már a z-shell-ben vagy. Igaz, nem nagyon néz ki másnak – nos, talán a prompt, hamar meg fogjuk változtatni –, de próbáld ki. Kétségtelenül ismered már a bash parancskiegészítését, de most írj be egy rendszeresen használt parancsot, de elfelejtetted az opcióit. Gépeljük be, hogy mount, üss egy szóközt, majd egy, vagy két kötőjelet és üss <Tab>-ot. Most a zshell kiegészítője megjeleníti a parancs összes opcióját (ha két kötőjelet írtál, akkor a hosszú opciókat), a leírásukkal. Az opció első egy, vagy két betűjét írd be, majd nyomj <Tab>-ot ismét, hogy a zshell befejezze helyettünk a gépelést.

A kiegészítést a zsh kifejezetten jól csinálja. Ha belenézel a .zshrc fájlba ezeket a sorokat találod

autoload -Uz
compinit compinit

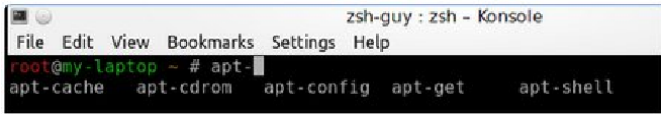
Ez azt jelenti, hogy a zsh.ban a kiegészítést beállító compinit modul betöltődött és végrehajtásra került.

Menüválasztás

Néha nagyon sok kiegészítési opciód van és a <Tab>-os léptetés közöttük nagyon fárasztó lehet. Ezért a zsh biztosít számunkra egy menü kiválasztási lehetőséget. Add a következő sort a .zshrc fájlod végéhez:

```
zstyle ':completion:*' menu select
```

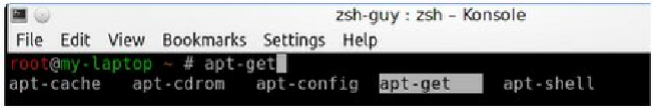
Most tegyük fel, hogy root-ként (feltételezve, hogy a root is zshell-t használ) telepíteni akarod a gnumeric táblázatkezelő alkalmazást, de nem vagy biztos a csomag pontos nevében, amit a csomagkezelőnek meg kell adnod.



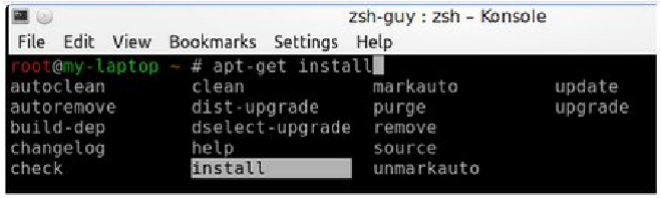
Írd be

apt- (tab)

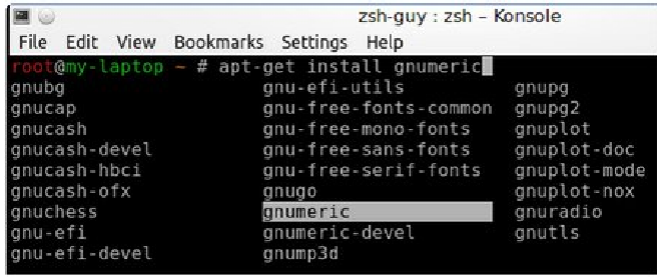
Látni fogod az elérhető parancsok listáját



Nyomj <Tab>-ot és az elsőt kijelöli. Most használd a nyíl gombokat az apt-get kiválasztásához.



Üss egy szóközt és a <Tab>-ot kétszer, hogy az alparancsok listáját megkapd, majd pozicionálj rá az install alparancsra.

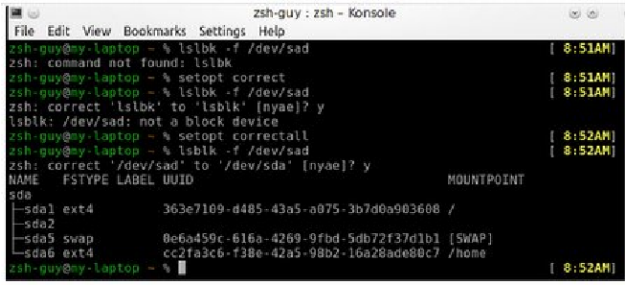


Most szóköz után írd be, hogy „gnu” és <Tab> kétszer, és mozgás le a gnumetric-hez, végül <Enter>. Az apt kilistázza az esetleges függőségeket, amiket szintén telepíteni, frissíteni kell, mielőtt a megerősítést kérné a folytatáshoz és telepítéshez.

Noha a zsh sok parancs sok kiegészítését ismeri, elkerülhetetlen, hogy találj olyan parancsot, aminek az opciói a zsh számára ismeretlenek. Ez esetben a zsh lehetővé teszi, hogy beprogramozd a saját parancskiegészítésedet szkript funkcióként, de ez talán meghaladja ennek a bemutatónak a kereteit.

Gépelési ellenőrzés

A gépelés ellenőrzésére két opció van, a correct és a correctall. Az első a parancsokhoz biztosít ellenőrzést, míg a második az argumentumokat vizsgálja. Nézd meg ezt a képernyőképet.



A parancs amit végre akartam hajtatni

```
lsblk -f /dev/sda
```

de mind az lsblk és a /dev/sda argumentum gépelési hibát tartalmaz. ENTER-t nyomva a shell szól, hogy nem ismeri az lslbk parancsot és megszakít. A „correct” beállítása esetén a parancsot újra hívva és az <Enter>-t lenyomva a shell felismeri a gépelési hibát és kérdezi, hogy javítsa-e ilyen formában

zsh: correct 'lslbk' to 'lsblk' [nyae]?.

„n” beírása azt jelenti, hogy nincs javítás. Estleg van egy olyan alias-od lslbk néven, amit a vizsgáló nem ismer. „y” azt jelenti, hogy lépjen tovább, cserélje, „a” azt jelenti, szakítsa meg a parancsot és az „e” a sorszerkesztést jeleníti meg, ha a javaslat nem tetszene.

„y”-t beírva a parancsot kiértékeli, de most azért panaszkodik, hogy a /dev/sad nem létezik. A „correctall”-t beállítva és újra próbálva a shell most felajánlja a /dev/sad javítását /dev/sda-ra. Az „y” kijavítja az argumentumot és a parancs rendben lefut.

A .zshrc-ben ezt a két setoption kifejezést beállítva mindez automatikusan történik meg.

Kötegelt átnevezés

Rendesen, ha fájlt akarsz átnevezni, az mv parancsot használod az egyik névről a másikra mozgatáshoz. A Unix/Linux alatt mindig ez volt a módja. Ha egy csomó fájlt kellett átnevezned, esetleg egy gyakori kiterjesztést, mondjuk a .zip-et .z-re, akkor az nem volt egyszerű.

Készíthettél egy rutint, hogy végigmenjen a fájlokon, azonosítva a lehetséges jelölteket, vagy használhattad a find-ot és az xargs-t, vagy az exec-t. Telepítetted akár az évek során mások, akik ugyanezen unalmas feladattal szembesültek, által írt eszközök egyikét. Ugyanakkor, ha zsh-t futtatsz használhatod a zmv-t.

A zmv alapból nem érhető el, az alábbi paranccsal kell betölteni

autoload -U zmv

Rendesen ezt a .zshrc fájlba kell beírni. A zmv alapvető használata egyezik az mv-vel. Vagyis az mv forrás cél helyett zmv forrás cél lesz.

Azonban van egy lényeges különbség. Ha dzsókert vezetünk be a forrásban (ismert még helyettesítő karakterként is), akkor a kiterjesztett helyettesítést alkalmazhatunk hivatkozásként a célon. A zip-es példánkra visszatérve így működik

```
zmv '(*).zip' '$1.z'
```

Az idézőjelek szükségesek a shell beavatkozásának megakadályozására addig, amíg a zmv parancs visszatartja, és a zárójelek a háttérhivakozás létrehozásához kell. Ha lennének ilyen fájljaink (kicsi az esélye, de talán...)

onetestcase.zip
twotestfile.zip
.....

és ilyesmire szeretnénk átnevezni

one_results_case.z
two_results_file.z
.....

Akkor egyszerűen adjuk ki a következő parancsot:

```
zmv '(*)test(*).zip' '$1_results_$2.z'
```

és az átnevezés megtörténik.

Testreszabás és prompt

Most akkor a prompt-ról. Add ezeket a sorokat a .zshrc-hez

```
autoload -U promptinit
promptinit
```

Így a promptinit modul betöltődik és végrehajtódik. A zsh egy sor előre meghatározott propmpt-tal érkezik, amiket ezzel átnézhetsz

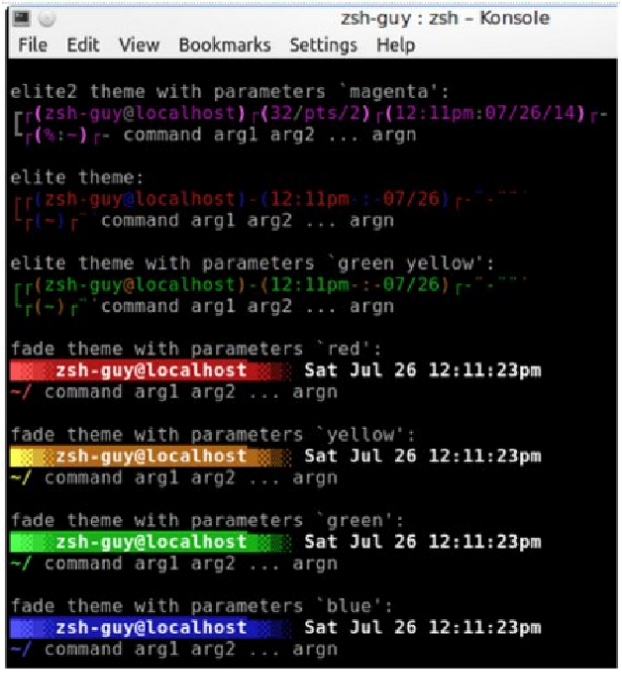
```
prompt -l
```

vagy előnézettel így

```
prompt -p
```

A terminálod hátterétől függően valami ilyen előnézeteket kapsz (következő lap, bal oldalt).

A prompt ezek egyikére állításához egyszerűen add a nevet és a paramétereket a .zshrc-ben a promptini inicializálása utánra. Például, a prompt-ot a fade



témára, kék paraméterrel állításhoz írd be ezt a beállító fájlba:

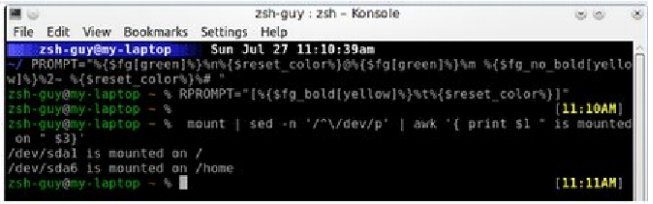
```
prompt fade blue
```

Megjegyzés: az alap prompt egyszerű felhasználóknak általában „$”, de a zsh-ban az alap a „%”. Én szeretem, mert emlékeztet, hogy zsh-t használok, nem bash-t. A prompt a kiemelt felhasználóknak (pl. root) továbbra is #.

Ha szertenél saját prompt-ot készíteni, az is rendben van. A zsh még azt is megengedi, hogy jobb, vagy bal igazítású pompt legyen, ahol a jobbra álló prompt automatikus eltűnik, ha az írás közelít. A színek kezelése a zsh-ban másképpen megy, tehát mielőtt színes prompt-ot készítenél azt add hozzá:

```
autoload -U colors
colors
```

a forrás fájlhoz.



A pompt beállítása „PROMPT=”-vel, a jobb prompt „RPROMPT=”-vel van.

A prompt egy szöveg és változó füzérből áll össze, csodálatosan le van írva az Arch Linux wiki-jében itt.

Vedd észre, hogy a jobbos prompt eltűnik az útból, ahogy hosszabb parancsot gépelsz. Amikor megfelelőnek találod, akkor mentsd a .zshrc-be, a szükséges funkció automatikus betöltése után.

Ha sokkal személyesebbet akarsz, akkor telepítheted a közösségek által készített keretek egyikét, pl. Robby Russell oh-my-zsh-ját (a git és wget is kell a PCLinuxOS tárolóiból)! Vissza a .zshrc-hez a telepítés előtt. Az oh-my-zshell több mint 120 témát, sok kiegészítőt, modult és alias-t tartalmaz, lehetővé téve a parancssor használatának átalakítását.

Nagyon nem javallt külső tárolókból telepíteni, de úgy vélem, ezzel a csomaggal kapcsolatban kivételt lehet tenni. Azonban ezt neked kell eldöntened. Ha ki akarod próbálni, írd be ezt egy terminálban.

```
Wget --no-check-certificate
https://github.com/robbyrussell/oh-my-
zsh/raw/master/tools/install.sh -O - | sh
```

Ez mindent beállít a gépeden. Indítsd újra a zsh-t, vagy jelentkezz ki, és be, majd kezdd olvasni a my-zsh docs-ot.

Kiterjesztett behelyettesítés

A fájlnevek helyettesítése fura meghatározás arra, amikor helyettesítő karaktereket, pl. * és ? kezdünk használni a nem ismertek helyett. A parancs

```
ls -ld D*
```

hosszú listát ad fel azokról a könyvtárakról, amik nagy „D” betűvel kezdődnek és 0, vagy több karakterrel folytatódnak. Meghatározhatunk tartományokat, osztályokat és karaktercsoportokat, vagy egész számokat. Noha mindez nagyon hasznos, a zsh ennél is tovább megy.

Nagyon hasznos lehetőség a rekurzív helyettesítés, amivel kereshetőek a megfelelő fájlok az adott és az alatti könyvtárakban.

```
ls **/.*rc
```

Ez minden rejtett beállító fájlt megkeres az adott könyvtárban. A kettős csillag állítja be mindezt. Ha szimbolikus linkeket is belevennéd, akkor használj

három csillagot ***- Ha a mintának megfelelők kivételével minden fájlt keresnéd, indítsd „^”-pal.

```
ls -d ^D*
```

Ez, a nagy „D”-vel kezdődő valamennyi könyvtárat megkeresi.

Másik hasznos funkció, amit a zsh keresője tartalmaz, az a minősítők. Ez egy, a keresési minta legvégén található zárójeles kifejezés, ami csak a minősített fájlok kijelzését engedélyezi.

```
ls -l /etc/*(#q@)
```

A példa megkeresi az összes szimbolikus linket a /etc könyvtárban. Használj „.”-ot a szokásos fájlokhoz, „/”-et a könyvtárakhoz, „@”-ot a linkekhez, „=” a socket-hez, „p”-t a csővezetékekhez. Sok van, sokkal többet tartalmaz a dokumentáció. Hasznos lehet még: * végrehajtható fájlokra és r, w, x a fájlok tulajdonos írási, olvasási és futtatási jogára.

Van még mód a közeli egyezésre a zsh-ban, ami megpróbálja a fájlokat egyeztetni, még ha gépelési hiba is lenne. Ez akkor hasznos, ha nem vagy teljesen biztos a cél nevében.

```
ls -d (#a1)Ducuments
```

helyesen megtalálja a Documents könyvtárat. A zárójelben szereplő 1 a szám, amennyi hibától eltekint, ám ha túl megengedő vagy, túl sok találatot fogsz kapni. Váltsd az 1-et 10-re és lásd, mire gondolok.

Többszörös átirányítás

Ez egyszerű és nagyon hasznos. A bash-ban ahhoz, hogy egy parancs kimenetét egyszerre megjelenítsd a terminálban és fájlba is küldd, a tee parancsot kell használnod, ami csak két adatfolyamra van korlátozva.

```
cat test_result | tee copy1
```

A zsh-ban ezt is megteheted:

```
<test_result >copy1 >copy2 >copy3
```

és annyi másolatot kapsz, amennyit szeretnél (nem tudom mi a limit). Ez csak egy bemenet és három kimenet. Ha a képernyőre is kiraknád, akkor csövezd a kimenetet egy záró cat parancshoz valahogy így:

```
<test_result >copy1 >copy2 >copy3 | cat
```

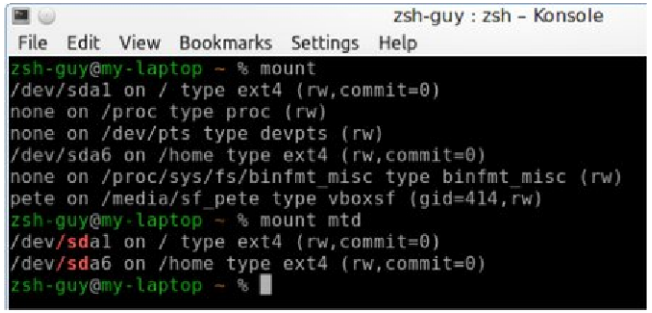
Azt hiszem ez sokkal logikusabb.

Globális alias-ok

Ez a bash-típusú shell-ek eleve hasznos alias-ának kiterjesztése. Majd' minden mehet a globális alias-ok közé és majdnem bárhol használható. Amikor a parancssorba beírom, hogy „mount” többet kapok, mint amire általában szükségem van. A kimentet tele van virtuális és nem hardver eredetű csatolási pontokkal, miközben én általában csak a csatoltnak jelölt rendszer-meghajtóimra vagyok kíváncsi.

A globális alias kap egy -g opciót és csak egy ilyen meghatározásom van

```
alias -g mtd='| grep /sd'
```



Vészre, hogy tartalmazhat függőleges vonal szimbólumot az alias meghatározásában.

Alias toldalékok

A legmodernebb asztali környezetek lehetővé teszik meghatározott fájltípusok hozzárendelését bizonyos alkalmazáshoz. A z-shell alias kiegészítési funkciójával ez parancssorban is lehetséges. Használd az alias-t -s opcióval, majd add hozzá a használandó alkalmazást. Íme néhány a z-shell wiki-jéből, amiket szeretek.

```
alias -s txt='less -rE'
```

Ezzel lehetőséged van .txt kiterjesztésű fájlok parancssorba történő beírását és <Enter>-t nyomva átnézheted a fájlt less-szel. Amikor kilépsz a parancssort kapod vissza.

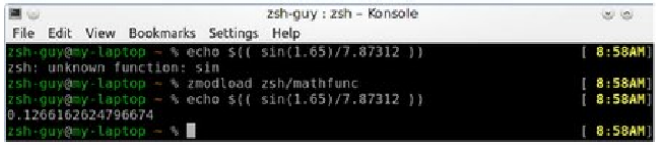
A következőt nagyon szeretem.

```
alias -s
html="/usr/share/vim/vim62/macros/less
.sh"
```

nyomj entert bármilyen .html fájlon és megnyílik less-szel, de a vim szintaxis kiemelése mellett.

Matematika

Lebegőpontos számolás alapból lehet a zsh-ban. Az egész számok úgy viselkednek mint a bash-ban. Az echo $((7/3)) eredményként 2-öt ad, miközben az echo $((7.0/3.0)), ami a bash-ban hibát ad, a zsh-ban 2.3333333333333335-at eredményez – elég jó megközelítés a legtöbb célra.



Sokkal komolyabb számításokhoz a bc, vagy a dc eszköz kellett korábban, ám a zsh nagyon jó modullal rendelkezik erre. Töltsd be ezzel a paranccsal

```
zmodload zsh/mathfunc
```

Egyáltalán nem kell elolvasnod a dokumentációt, hogy rájöjj, ez neked való, mivel majdnem mindent kezel, amire egy nem-Einsteinnek szüksége lehet.

Auto CD

Egy, aminek aktiválását javaslom az indító beállításokban a „váltás egy adott könyvtárra az útvonal megadásával”. Ez hozzáadja a „setopt autocd”-t a .zshrc-dhez. Nagyon hasznos, hiszen a cd parancs a leggyakrabban használt parancs terminálban – most már majdnem teljesen elfelejtheted!

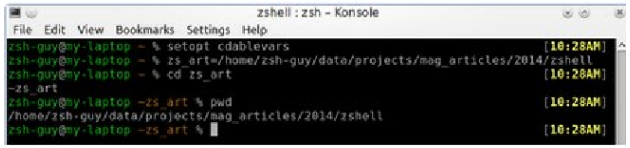
Ezt az opciót beállítva egyszerűen gépeld be a könyvtár nevét, hogy oda menj.

```
/etc belép                  /etc-be;
~           belép a home
            könyvtárba;
..          egy könyvtárat fellép;
/usr/share/icons/locolor
            belép abba a könyvtárba;
../hicolor  a /usr/share/icons/hicolor
            könyvtárra vált
```

A könyvtár nevének gépelését segítő <Tab>kiegészítés természetesen működik.

További opció, ami hasznos lehet a könyvtárútvonalak elnevezése, ami ezzel a paranccsal aktiválható

```
setopt cdablevars
```



Ha van egy könyvtárad, amit rendszeresen használsz, de eléggé el van ásva a könyvtárszerkezetben, akkor beállíthatsz egy oda mutató változót. Ezután cd a változóra és igen, itt be kell gépelned a cd-t.

A változó neve kerül a prompt-ba, tehát gondoskodj legalább számodra értelmes változónévről. Úgy vélem a képernyőkép prompt időbélyegéből látszik, mennyire javítja teljesítményt a kiegészítés és automatizálás és én még nem is vagyok gyors gépelő.

A könyvtár-verem használata

Többségünknek, ha nem mindnyájunknak van valamilyen eszköze a könyvtár-vermek kezelésére, de én még sosem láttam ilyen rugalmas rendszert, mint amilyen a zsh-é. A bash a pushd-t és a popd-t használja jó hatásfokkal, de könnyen el lehet tévedni, amikor gyorsan váltunk sok könyvtár között. Az zsh előnye, hogy lehetővé teszi a vermek közvetlen manipulálását. Ez a hagyományos pushd és popd parancsok kiegészítése.

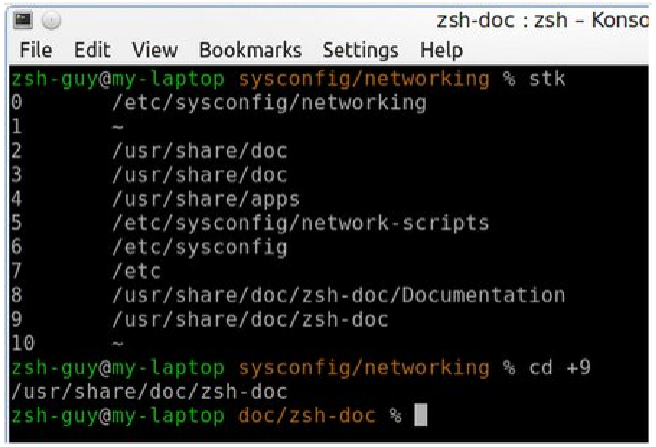
Egy hosszú terminálfolyamatban tucatnyi könyvtárat látogatok meg, néhány mélyen el van ásva az alkönyvtárakban é s gyakran újból belépek egyesekbe másik, messzi könyvtárból.

Az autopushd opció beállításával ez sokkal könnyebb. Minden egyes könyvtár, amit meglátogattam a verembe kerül és az aktuális verem megtekinthető a következő paranccsal:

```
dirs -v
```

Nagyon lusta vagyok, még erre a rövid parancsra is alias-t használok:

```
alias stk='dirs -v'
```



Most egyszerűen válthatsz a megszámozott könyvtárba. Használd a „cd +n”, vagy „cd -n” a veremben fentről, vagy lentről számolva n. könyvtárhoz. Íme egy rövid bemutató a működéséről.

És sok egyéb...

Igen, több is van. A bemutató alig súrolta csak a felszínt, de remélhetőleg eleget mutat ahhoz, hogy ráérezz a zsh ízére. Noha sok minden lehetséges a zsh-val, de használható a bash-hoz hasonló módon, vagy csak az általad kedveltekkel kiegészítve.

A dokumentáció jó, noha egy kicsit száraz. Számos kézikönyvlap és még függvények is vannak benne, hogy segítsék neked a dolgokat megtalálni és még tonnányi infó található az Interneten is.