Kódolás és IT biztonság (VIHIBB01)
Laboratóriumi gyakorlat

1. labor - Linux alapok (ism.)

Szerzők:
Gazdag András
Holczer Tamás
Ládi Gergő

2023. szeptember 18.

Tartalomjegyzék

1. A labor célja

A labor gyakorlat során a Linux operációs rendszer alapvető működésével fogsz megismerkedni. A cél, hogy az operációs rendszer felépítését és működését megértsd, valamint, hogy a leggyakrabban használt programok alapjait megtanuld.

A munka során egy NGINX szerver telepítését és konfigurálását fogod elvégezni. A labor végére egy működőképes és helyesen bekonfigurált webszervernek kell elkészülnie.

2. Háttéranyag

A tárgy, és a hozzá tartozó laborok építenek az Operációs rendszerek tárgyra és az ott elsajátított tudásra. Az első labor részben bevezető, részben ismétlő céllal készült. A laborhoz szükséges legfontosabb háttérismeret szerepel ebben a fejezetben, azonban ez csak egy összefoglaló, a feladatokhoz a korábban megszerzett alaposabb tudás is szükséges.

2.1. A Linux rövid története

A Linux operációs rendszer valószínűleg a legismertebb nyílt forráskódú szoftver. Az operációs rendszer lelkét jelentő kernelt Linus Torvalds kezdte fejleszteni 1991-ben. Ehhez kapcsolódik még a GNU projekt keretében fejlesztett programcsomag, amely így együtt egy jól használható operációs rendszerré áll össze. Több kiadása is létezik a Linux operációs rendszernek, amelyeket disztribúcióknak nevezünk. A legnépszerűbb disztribúciók közé tartozik például az Ubuntu, a Debian, Red Hat Enterprise Linux, vagy az Arch Linux. A félév során a laborok feladatait egy Ubuntu rendszeren kell elkészíteni.

2.1.1. Csomagkezelés

Linux operációs rendszeren hagyományosan csomagkezelő segítségével lehet programokat telepíteni. Ez a szoftverterjesztési módszer lényegében megegyezik a mobil világban is elterjedt app store megközelítéssel. A programok különböző repositorykban (csomagtárolók) találhatóak, melyek használata megkönnyíti a csomagok és függőségeik kezelését. Ubuntu és Debian rendszerek esetében javarészt három megoldással találkozhatunk: `dpkg`, `apt`, és `apt-get`.

Telepítés `dpkg`-vel. Amennyiben közvetlenül hozzáférünk a csomaghoz (pl: megvan már lokálisan vagy USB eszközön), akkor használhatjuk a telepítésre a hagyományos módszert, a `dpkg`-t. Ennek segítségével a `dpkg -i <csomag>.deb` utasítással tudunk telepíteni. A nehézség ezzel a módszerrel, hogy ez nem kezeli a függőségeket, azok telepítéséről külön, kézzel kell gondoskodnunk. Ezért kényelmesebb az `apt` (vagy régebbi rendszereken az `apt-get`) használata.

Telepítés `apt`-vel. Kényelmesebb és modernebb megoldás a csomagtelepítésre az `apt` használata. Ennek segítségével két lépésben szokott zajlani a telepítés:

1. `apt update` - a csomaglista frissítése
2. `apt install <csomag>` - a csomag tényleges telepítése

A már telepített csomagok az `apt upgrade` paranccsal frissíthetők.

Hasznos kiegészítés a korábbiakhoz, hogy az `apt-get` lehetőséget ad arra is, hogy csak letöltsünk egy csomagot, de azt ne telepítsük. Ezt a `apt-get install --download-only <csomag>` parancs segítségével tudjuk megtenni. Ezt a módszert mi is alkalmaztuk a laborok virtuális gépeinek a készítésekor, hogy a feladatok megoldása során a telepítési lépéseknél kevesebb csomagot kelljen letölteni, és így hatékonyabb és gyorsabb legyen a munkavégzés.

Érdemes továbbá megjegyezni, hogy ezeken felül is léteznek csomagkezelő megoldások, például a `yum` vagy a `zypper`, valamint amennyiben rendelkezésre áll egy csomag forráskódja, úgy van lehetőségünk forrásból való telepítésre is.

2.2. Rendszer felépítése

A labor során használt rendszer *ext* fájlrendszert használ. A fájlrendszeren található könyvtárstruktúra ismerete szükséges az operációs rendszer használatához. A legfontosabb könyvtárak és funkciójuk a következőek:

- `/` - a fájlrendszer gyökere, minden fájl elérése innen indul ki.
- `/bin` - futtatható binárisok helye

- `/boot` - az operációs rendszer betöltéséhez szükséges fájlok (pl: GRUB, kernel, stb.)
- `/dev` - hardverelemek elérése a fájlrendszeren keresztül
- `/etc` - globális konfigurációs fájlok helye
- `/home` - felhasználók saját mappái és fájljai
- `/lib` - megosztott könyvtárak és kernel modulok helye
- `/lost+found` - sérült és/vagy elveszett fájlok kerülnek ide
- `/mnt` - felcsatolt médiaeszközök (pl: USB eszközök)
- `/proc` - rendszerinformációk, valamint az éppen futó folyamatokról elérhető információk
- `/root` - a root felhasználó saját könyvtára
- `/run` - az indítás során elérhető tmpfs fájlrendszer, amelyre néhány programnak szüksége van az elinduláshoz (pl: udev)
- `/sys` - futás közbeni információk lekérdezésére és beállítására alkalmas hely a kernel számára (`/proc`-hoz hasonlít, csak ez a kernelhez van)
- `/usr` - felhasználó által telepített programok, könyvtárak, dokumentációk és forráskódok helye
- `/var` - gyakran változó adatok helye, átmeneti tárolók (cache), naplófájlok gyűjtőhelye

2.3. Alapvető parancsok

Egy Linux rendszer használata során nagyon sok feladatot el lehet végezni a rendszerben alapból elérhető binárisok segítségével. Ezek ismerete jelentősen felgyorsítja a használatot. A legtöbb alkalmazásnak elég csak a nevét begépelni, és az operációs rendszer tudni fogja, hogy az hol található, és így el fogja tudni indítani (ez főképp az előre jól definiált fájlrendszerstruktúrának köszönhető). A valódi helyzet még ennél is kényelmesebb, hiszen ha egy program nevének begépeljük az első pár betűjét, akkor a TAB billentyű kétszeri lenyomása hatására kapunk egy listát az elérhető programokról, amelyek neve a begépeltnek megfelelően kezdődik. Ezt a név kiegészítő funkcióját a rendszernek minden parancs kiadásakor javasolt használni, mert így sokkal kisebb az esély arra, hogy egy hibás parancsot adunk ki, ami veszélyes lenne. Nemcsak a programok nevét, hanem az adott helyeken könyvtárak és fájlok nevét is ki tudja egészíteni a rendszer, amelyek teljes begépelésére így szintén nincs szükség.

A leggyakrabban használt programok és parancsok közül néhány:

- `ls <könyvtár>` - fájlok kilistázása a megadott könyvtárból; gyakran használt kapcsolója a `-l`, ami részletesebb információt ír ki
- `pwd` - aktuális munkakönyvtár elérési útjának a kiírása
- `cd <könyvtár>` - aktuális munkakönyvtár megváltoztatása
- `mv <forrás> <cél>` - fájlok vagy könyvtárak mozgatására és átnevezésére használt parancs
- `cp <forrás> <cél>` - fájlok vagy könyvtárak másolására használt parancs
- `nano <fájlnév>` - szövegszerkesztő program
- `vi <fájlnév>` - szövegszerkesztő program
- `vim <fájlnév>` - szövegszerkesztő program
- `cat <fájlnév>` - a megadott fájl tartalmát kiírja a konzolra
- `less <fájlnév>` - szöveges fájlok csak olvasására használt program
- `rm <fájlnév>` - fájl vagy könyvtár törlésére használt program
- `mkdir <könyvtárnév>` - új könyvtár létrehozása a megadott néven
- `ln <forrás> <cél>` - szimbolikus link létrehozására alkalmas, a `-s` kapcsolóval soft linket hoz létre
- `sudo <parancs>` - rendszergazdai jogosultsággal lehet egy parancsot futtatni (alapból nincs telepítve)
- `su <felhasználónév>` - felhasználó váltására alkalmas; amennyiben nincs megadva felhasználónév, akkor a root felhasználóra lehet váltani vele
- `<parancs> | <parancs>` - pipe (cső) segítségével az első parancs kimenete továbbítható egyből a második parancs bemenetére

2.3.1. Szövegszerkesztők használata

Egy rendszer beállítása során az egyik leggyakrabban előforduló feladat egy konfigurációs fájl szerkesztése. Erre a feladatra NEM alkalmasak a széles körben elterjedt szövegszerkesztők (Microsoft Word, vagy Libre Office). Ezek az alkalmazások egyedi fájlformátumot használnak, így amennyiben valamelyiken keresztül készül egy konfigurációs fájl, azt nem fogja tudni megnyitni az azt használó alkalmazás.

Konfigurációs fájlok szerkesztésére csak a rendszer beépített, vagy ilyen célra is fejlesztett egyszerűbb szövegszerkesztők használhatók, úgy mint a

`nano`, a `vi`, vagy a `vim`. Ezek használata során csak azok a karakterek kerülnek bele az elmentett fájlokba, amelyeket ténylegesen begépelünk, így nem fogunk kompatibilitási problémákba ütközni. A felsoroltak közül a `nano` használata valószínűleg a legegyszerűbb, azonban egyéni preferencia alapján mindegyikre van lehetőség. Néhány helyen a rendszer automatikusan valamelyik szövegszerkesztővel nyit meg egy fájlt, így az alapszintű ismerete mindhárom programnak szükséges[1].

2.4. Jogosultságkezelés

A felhasználó- és jogosultságkezelés egy külön labornak a témája, azonban a korábbi laborokon is szükséges néhány alapkoncepció alkalmazása.

Linux alatt alapvetően háromféle jogot különböztetünk meg: olvasási, írási és végrehajtási jog. Ezeket a jogokat háromféle kategóriában osztjuk ki: a tulajdonosnak, egy csoportnak, és mindenki más számára. Minden újonnan létrejövő fájl egy alapbeállítással jön létre a jogosultság szempontjából is, ez pedig a következő:

- a tulajdonos olvashatja és írhatja is a fájlt
- a tulajdonos csoportja csak olvashatja a fájlt
- mindenki más szintén csak olvashatja a fájlt

A konfigurációs fájlok, amelyek a `/etc` mappában találhatóak, rendszerint a root felhasználó tulajdonában állnak. Amennyiben egy konfigurációs fájl használata során abba a hibaüzenetbe ütközünk, hogy nincs jogosultságunk megnyitni vagy elmenteni egy konfigurációs fájlt, mindig elsőként a fájlrendszeren beállított jogosultságokat ellenőrizzük. Ha szükséges, akkor váltsunk felhasználót, ami általában meg is oldja a problémát.

2.5. Szolgáltatások

Minden rendszeren futnak olyan folyamatok is a háttérben, melyek nem feltétlenül szükségesek az operációs rendszer működéséhez, és nem is egy konkrét felhasználó indította el őket. Ezek tipikusan szolgáltatások (*service*), melyek feladata valamilyen funkcionalitás biztosítása a helyi vagy távoli felhasználók számára. Ilyen szolgáltatás lehet például távoli bejelentkezés (SSH,

[1] https://stackoverflow.com/questions/11828270/how-do-i-exit-the-vim-editor, https://github.com/hakluke/how-to-exit-vim

Remote Desktop) támogatása, nyomtatók és nyomtatandó dokumentumok kezelése vagy weboldalak kiszolgálása.

A szolgáltatások a beállításaiktól függően általában az operációs rendszer betöltődése után automatikusan elindulnak, de adott esetben szükség lehet ezek kézi leállítására, újraindítására, ellenőrzésére. Az ehhez hasonló feladatok elvégzéséhez szolgálnak segítségünkre az alábbi parancsok:

- `service --status-all` - az összes telepített szolgáltatás listázása azok állapotával együtt
- `service <név> status` - adott szolgáltatás állapotának lekérdezése
- `service <név> start` - adott szolgáltatás elindítása
- `service <név> stop` - adott szolgáltatás leállítása
- `service <név> restart` - adott szolgáltatás újraindítása
- `service <név> reload` - jelzés küldése egy szolgáltatásnak, hogy töltse újra a konfigurációs fájljait

Nem minden szolgáltatás támogat minden műveletet. A listázás és állapotlekérdezés kivételével ezen parancsok futtatásához rendszergazda jogosultság szükséges.

3. Feladatok

3.1. Vezetett rész

A labor elején a laborvezető segítségével közösen megismerkedtek a félév során használandó virtuális környezettel. Ezt követően átismétlitek a Linux rendszerekkel, azok kezelésével kapcsolatos alapvető tudnivalókat: megnézitek a fájlrendszer felépítését, elvégeztek pár fájlokkal kapcsolatos műveletet, majd telepítetek és beállítotok egy szolgáltatást.

Részletes leírás hallgatóknak

1. Indítsd el a virtuális géped. Ha még nincs, nézd meg és kövesd az *általános tudnivalók* című segédletet (link a Moodle-ben).
2. Indíts egy terminált. Nézd meg, mi a munkakönyvtárad (*pwd*).
3. Nézd meg és értelmezd egy *ls* és egy *ls -la* kimenetét ott, ahol állsz, majd nézd meg a gyökeret (*ls -la /*). Miket látsz itt? Melyik könyvtár mire szolgál?
4. A gépre telepítve van egy Juice Shop nevű komplett webalkalmazás kliens- és szerveroldali résszel. Ez szándékosan számos biztonsági problémát tartalmaz, amelyeket ki is fogsz használni a következő laborokon. A mai alkalommal ennek az alkalmazásnak készíted elő a terepet, így javasolt legalább a vezetett rész végére érned, különben nehézségek adódhatnak a későbbi laborok során. A Juice Shop jelenleg nem fut, el kellene indítani. Ehhez először lépj át a */home/cloud/juice-shop* könyvtárba a *cd /home/cloud/juice-shop* paranccsal, majd indítsd el az alkalmazást az *npm start* paranccsal. Az indulás a gépek terheltségétől függően pár percig is eltarhat, de mivel a következő néhány feladatponthoz nem lesz szükség az alkalmazásra, folytathatod a feladatok megoldását.
5. A *juice-shop* interaktív módban fut, így az eddig használt terminálablakba nem fogsz tudni további parancsokat gépelni. Indíts egy új terminált, ügyelve arra, hogy az előzőt ne zárd be.
6. Az nginx egy lightweight webszerver, melyet gyakran használunk reverse proxyként (egy másik szolgáltatás elé előtétként, pl. biztonságot nyújtva) vagy terheléselosztóként is. A mai laboron egy ilyet szeretnénk telepíteni és betenni a Juice Shop elé. Telepítsük az nginxet: *apt install nginx*. Nem fog sikerülni. Próbáld meg kideríteni a hibaüzenet alapján, hogy miért.

7. Ugyanezt ismételd meg rootként: *sudo apt install nginx*. Bátrabbak megpróbálhatják a *sudo !!* parancsot is. (Ez mit is csinál?) Értelmezd a látottakat.
8. Nézd meg, jelenleg fut-e az nginx (*service nginx status*). Ha nem, akkor indítsd el a *service nginx start* paranccsal, egyébként nyisd meg a böngészőt. Navigálj el a http://juice-shop.crysys.hu oldalra, ahol egy default nginx kezdőlapnak kell bejönnie egyelőre. (Hogyan lehetséges, hogy mindenkinek a saját gépén futó nginx szerver válaszol a cím beírása után?)
9. Nyisd meg a http://127.0.0.1:3000 címet. Ha mostanra elindult a Juice Shop, betölt egy weboldal. A weboldalra ezen a laboron nem lesz szükség, de elérhetőnek kell lennie. Előfordulhat, hogy még nem indult el az oldal, ebben az esetben nézz rá a labor elején indított terminálra. A konzolon az utolsó sorban a *Server listening on port 3000* szövegnek kell látszódnia.
10. Tetszőleges szövegszerkesztővel nyisd meg a */etc/nginx/nginx.conf* fájlt. Itt a webszerver általános beállításai vannak. Tanulmányozd a fájlt. Mit jelenthet az *include /etc/nginx/sites-enabled/** sor? Egy webszerver több weboldalt (pongyolán fogalmazva "domaint") is ki tud szolgálni, és ezekhez mind külön-külön konfigurációs paraméterek tartoz(hat)nak. Ezeket ezen az útvonalon keresi. Oda kell majd tenned valahogy a te beállításaidat is.
11. Nézz rá a */etc/nginx/sites-enabled/* útvonalra *ls*-sel: *ls -la /etc/nginx/sites-enabled/* (a kapcsolók fontosak!). Látható, hogy van egy *default* nevű konfig fájl, ami egy symlink a */etc/nginx/sites-available/default*-ra (mi az a symlink?). Majd neked is valami ilyesmit kell csinálni. Ennek a jelenségnek az oka, hogy a sites-available könyvtárban gyűjtünk minden konfigot, majd a ténylegesen használni kívántakat átsymlinkeljük a sites-enabled-be. Így a tényleges konfig fájlok törlése nélkül könnyen tudunk ki-be kapcsolni egy-egy site-ot.
12. Hogy a Juice Shopra vonatkozó konfig fájlt ne a nulláról kelljen megírni, készült egy szkeleton. Ennek a linkje megtalálható a Moodle-ben. Töltsd le *wget*-tel.
13. Szerkeszd a letöltött fájlt valamilyen szövegszerkesztővel, értelmezd a tartalmát, majd a kommentek alapján töltsd ki. Ehhez hasonlóan kell kinéznie:

```
server {
```

```
    # Listen on the default HTTP port
    listen 80;

    # Apply these settings to traffic directed at juice-shop.crysys.hu
    server_name juice-shop.crysys.hu;

    location / {
        # Forward any incoming request to juice-shop at http://localhost:3000
        proxy_pass http://localhost:3000;

        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header Host $host;
    }
}
```

14. Mozgasd át a fájlt a */etc/nginx/sites-available* útvonalra, *juice-shop.crysys.hu.conf* néven (*mv*).
15. A most átmozgatott konfig fájlt symlinkeld be a *sites-enabled*-be: *ln -s /etc/nginx/sites-available/juice-shop.crysys.hu.conf /etc/nginx/sites-enabled/juice-shop.crysys.hu.conf*. Fontos a sorrend. *ls -la*-val megnézheted, sikerült-e. A *-s* kapcsoló miatt soft link készül (ez egy alfajtája a symlinknek), ami érvénytelenné válik, ha a hivatkozott fájlt törlik.
16. Szólj az nginxnek, hogy töltse újra a konfig fájlokat: *service nginx reload*. Ha hibát kapsz, valamit elgépeltél. Javítás után *service nginx start* kell.
17. Nézd meg a http://juice-shop.crysys.hu oldalt. Be kell töltenie.
18. *more*, *less*, *tail*, esetleg *cat* paranccsal nézd meg a webszerver logjait a */var/log/nginx/access.log* fájlban. Látszódnia kell az előző oldalbetöltésnek. Próbáld ki a *tail -f* parancsot is a logfájlon! Az mit csinál?

3.2. Önálló feladatok

3.2.1. 1. feladat - További nginx konfiguráció

Szeretnénk, hogy a Juice Shoppal kapcsolatos események ne az előbbiekben vizsgált, ömlesztett naplófájlba kerüljenek, hanem külön helyre, */var/log/nginx/juice-shop/access.log* és */var/log/nginx/juice-shop/error.log* néven. Hozd létre a könyvtárat az új naplófájloknak, majd állítsd be az nginxet, hogy ezek (és csak ezek!) a naplófájlok a megadott helyre kerüljenek. Munkádat ellenőrizd! Amennyiben nem a várt működést tapasztalod, próbáld meg kitalálni, mi a probléma, és próbáld is meg megoldani azt!

3.2.2. 2. feladat - Nincs itt semmi látnivaló

Próbálj meg megnyitni néhány olyan aloldalt böngészőből az nginx default site-ján belül (tehát nem a Juice Shopon belül!), amely biztosan nem létezik (például: http://127.0.0.1/asdf123). Nézd meg a megfelelő naplóban, milyen státuszkóddal jelennek meg ezek a lekérések. Adj meg egy parancsot, mellyel ki tudjuk listáztatni a naplófájlból csak azokat a sorokat, melyekben szerepel ezen státuszkód értéke (az most mindegy, hogy pontosan hol, csak valahol szerepeljen a sorban).