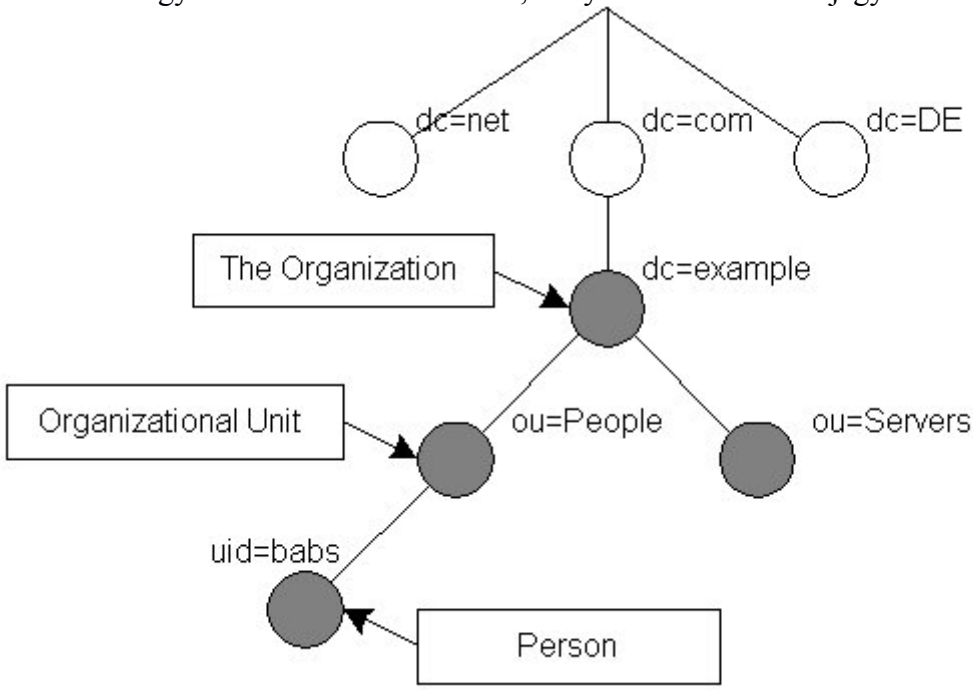
3. Gyakorlat

LDAP:

- Lightweight Directory Access Protocol
- Directory szolgáltatások elérését szabályozó protokoll.
- A directory szolgáltatás egy keresésre optimalizált speciális adatbázist takar, amelyben az információ egy fa szerkezetben tárolódik, melynek csúcsaiban bejegyzések szerepelnek.



- Egy bejegyzésre egyértelműen hivatkozhatunk a DN-jével (Distinguished Name), ami a fában a csúcshoz vezető utat írja le.
- Az ábrán látható példában a Person DN-je: uid=babs,ou=People,dc=example,dc=com
- A fában való kereséshez az **ldapsearch** program használható, ehhez a keresés kezdőpontját (base), hatáskörét (scope) és egy szűrőt (filter) kell megadni.
- A **base** egy DN, ahol a keresést kezdeni kell, csak az ez alatti bejegyzéseket veszi figyelembe a keresés.
- A **scope** a base, one, sub és children értékek valamelyikét veheti fel értékül.
  - base: csak a base DN által megadott bejegyzés szerepel a keresés hatáskörében.
  - one: a base DN és az egy szinttel lejjebb lévő bejegyzések tartoznak a keresés hatáskörébe.
  - sub: a base DN és az alatta lévő részfa tartozik a keresés hatáskörébe, ez az alapértelmezett értéke a scope-nak.
  - children: a base DN gyerekeire terjed ki a keresés hatáskörébe.
- A **filter** a kereséshez megadható szűrő, a bejegyzések attribútumaira szűrhetünk.
- Példák:
  - ldapsearch –x -LLL uid=$USER
  - ldapsearch –xLLL –b ou=People,ou=users,dc=iit,dc=uni-miskolc,dc=hu –s one uid=$USER
  - ldapsearch –xLLL cn=”*Alex” | grep cn | cut -d " " -f 2 | sort –u
  - ldapsearch –xLLL homeDirectory | grep homeDirectory | cut –d’/’ –f3 | sort –u

I/O műveletek:

- stdout átirányítása fájlba: >
- stdout átirányítása fájlba, hozzáfűzéssel: >>
- stderr átirányítása stdout-ra: 2> &1
- stdin átirányítása: <
- pipeline: |
- Speciális fájlok:
  - /dev/null
    - Bármi, amit beleírunk, eltűnik, olvasáskor rögtön EOF-ot ad.
    - Hibaüzenet eltüntetése: cd nem_létező_jegyzék 2> /dev/null
    - Üres fájl létrehozása: cat /dev/null > új_fájl_neve
  - /dev/zero
    - Bármi, amit beleírunk, eltűnik, olvasáskor \0 karaktereket ad.
    - Adott méretű fájl létrehozása: dd if=/dev/zero of=fajl count=1024 bs=1

Shell scriptek:

- Olyan szöveges állományok, melyek UNIX parancsokat tartalmaznak.
- A fájl első sora egy parancsértelmező megadását tartalmazza (pl.: #!/bin/bash), ha nincs megadva, akkor a scriptet az alapértelmezett parancsértelmező hajtja végre ($SHELL).
- A parancsok magukban foglalhatnak különböző külső- és belső parancshívásokat, szelekciós, iterációs, stb. utasításokat.
- A scriptben használhatunk előre definiált változókat, valamint létrehozhatunk sajátokat.
  - Előre definiált változók pl.:
    - Környezeti változók pl.: `SHELL`, `PWD`, `PATH`, `USER`, stb.
    - Pozicionális paraméterek pl.: `$0`, `$1`, ..., `$9`, `$#`, `$*`, `$@`, `$$`, `$?`, stb.
  - Saját változók létrehozása: `változónév=érték`
- A változók értékére $változónév formában hivatkozhatunk.
- Szelekciós utasítások:
  - **if szerkezet:**

```
if kifejezés
then
 utasítások
elif kifejezés
 utasítások
else
 utasítások
fi
```

  - **case szerkezet (switch):**

```
case string in
 minta1)
    utasítások;;
 minta2)
    utasítások;;
 *)
    utasítások;;
esac
```

- Iterációs utasítások:
  - **for ciklus:**
    ```
    for ciklusváltozó in lista
    do
     utasítások
    done
    ```
  - **while ciklus:**
    ```
    while kifejezés
    do
     utasítások
    done
    ```
  - **until ciklus:**
    ```
    until kifejezés
    do
     utasítások
    done
    ```
- Függvények: `function függvénynév { utasítások }`

Export:

- A létrehozott változók ahhoz a shellhez tartoznak, amelyben létrehozzuk őket, és csak ebben elérhetők, az export paranccsal azonban elérhetővé tehetők a shell gyerekprocesszei számára.
- Vegyük a következő példát:
  - Létrehozunk egy változót: `var=érték`
  - Indítunk egy újabb shellt: `sh`
  - Kiíratjuk a változó értékét: `echo $var`
  - Az eredmény egy üres sor, a változó értéke nem elérhető.
  - Megoldás: az újabb shell indítása előtt az export segítségével a gyerekprocesszek számára elérhetővé kell tenni a változót: `export var`
- Használata:
  - `var=érték && export var`
  - `export var=érték`

Meta karakterek, reguláris kifejezések:

- A reguláris kifejezések meta karakterekből tevődnek össze, karakterminták írhatók le
- velük, illeszkedést vizsgálhatunk a segítségükkel.
- Meta karakterek:
  - „`.`” bármely 1 karakterre illeszkedik
  - „`?`” 0 vagy 1 karakterre illeszkedik
  - „`*`” 0 vagy több karakterre illeszkedik
  - „`[...]`” a zárójelben megadott karakterekre illeszkedik (pl.: `[abc][0-9]`)
  - „`[^...]`” bármely 1 karakterre illeszkedik, ami nincs a zárójelben
  - „`^...`” sor elején illeszkedés, „`...$`” sor végén illeszkedés
  - „`\<...`” szó elején illeszkedés, „`...\>`” szó végén illeszkedés
  - „`+`” a megelőző kifejezés 1 vagy többszöri ismétlődése
  - „`{N}`” a megelőző kifejezés N számú ismétlődése
  - „`{N,M}`” a megelőző kifejezés min. N, max. M számú ismétlődése

- Példa, telefonszámok: „`+36-[237]0-[0-9]\{3\}-[0-9]\{3,4\}`”
- Példa, személyi igazolványszámok: „`[0-9]\{6\}[A-Z]\{2\}`”
- Példa, email címek: „`[a-zA-Z0-9_.]\+@[a-z0-9]\+.[a-z]\{2,\}`”

Feladat:

- Írjunk scriptet, amely egy egyszerű TODO alkalmazást valósít meg!
- Lehessen megjeleníteni a TODO lista bejegyzéseit, hozzáadni, valamint törölni bejegyzéseket!
- A bejegyzések egy címet, leírást és egy dátumot tartalmaznak.
- A beolvasott adatokra legyen ellenőrzés!