Od początku system Unix projektowano tak, by mogło na nim pracować wielu użytkowników w tym samym czasie. Aby zachować prywatność i odseparować jednego użytkownika od drugiego zaimplementowano system własności i uprawnień. Cały ten system jest dość prosty, choć oczywiście dla początkujących może wydawać się skomplikowany i enigmatyczny. Oparty jest on na tym, że każdy plik w systemie określone ma trzy parametry: właściciela, grupę i zestaw praw dostępu. Ponieważ w Unixie wszystko jest plikiem, w szczególności także proces, więc cały ten system uprawnień dotyczy wszystkiego co mamy w Unixie.

Spis użytkowników wraz z podstawowymi ich danymi znajduje się w pliku `/etc/passwd`. Nazwa pochodzi stąd, że historycznie znajdowały się tam hasła. Zrezygnowano z tego ponieważ plik ten musi być czytelny dla wszystkich i nawet jeśli przechowywane hasło byłoby zaszyfrowane to kusiłoby aby je złamać. Teraz w miejscu hasła znajduje się znak `x`. Hasła przechowywane są w pliku `/etc/shadow`, do którego dostęp jest mocno ograniczony. Są tam również informacje o tym kiedy hasło było zmieniane, jak długo jest aktualne, kiedy wymusić jego zmianę itp.. Pojedynczy wpis w pliku `/etc/passwd` dla zwykłego użytkownika może wyglądać tak:

```
mariusz:x:1001:10:Mariusz Zynel:/export/home/mariusz:/bin/bash
```

Jak widać znakiem `:` oddzielone zostały poszczególne dane, które mają następujące znaczenie:

```
username:password:uid:gid:gcos-field:home-dir:login-shell
```

Komputery lepiej sobie radzą z liczbami niż napisami, więc każdej nazwie użytkownika (zwanej potocznie *login*, ale formalnie login to proces logowania do systemu) przypisany jest identyfikator liczbowy UID. Oczywiście i nazwa użytkownika i jego UID (user identifier) są unikalne. Dla ułatwienia zarządzania prawami dostępu użytkowników można łączyć w grupy. W pliku `/etc/group` znajduje się lista grup. Pojedynczy wpis wygląda tak:

```
sys::3:root,bin,adm
```

Znowu znak `:` został zastosowany by rozdzielić 4 pola:

```
groupname:password:gid:user-list
```

W pierwszym polu znajduje się nazwa grupy, a w trzecim identyfikator liczbowy GID (group indentifier). One też muszą być unikalne. W ostatnim polu umieszczona jest lista użytkowników należących do tej grupy jako drugorzędnej. Jeden użytkownik może być członkiem wielu grup, ale tylko jedna z nich jest podstawowa (primary) - ta określona w `/etc/passwd` poprzez GID. Pozostałe grupy, których jest członkiem, to jego grupy drugorzędne (secondary) - do tych przypisany jest w pliku `/etc/group`. W powyższym przykładzie do grupy `sys` należą użytkownicy: `root`, `bin` oraz `adm`.

Mówiąc o przywilejach w Unixie nie można zapomnieć o specjalnym użytkowniku, który może w systemie wszystko. Jego nazwa to `root` a jego UID to 0. Jest również grupa o tej samej nazwie, której GID to 0. Synonimem użytkownika `root` używanym w żargonie jest *superuser*. To chyba dobitnie wyraża jego możliwości i przywileje.

Po zalogowaniu się do systemu można przełączyć się na innego użytkownika o ile tylko znamy jego hasło. Służy do tego polecenie `su`, bardzo często błędnie rozumiane jako *superuser*. Poleceniem `su` (*switch user*) możemy przełączyć się na dowolnego użytkownika systemu podając jego nazwę jako parametr. Jeśli tego parametru nie podamy to domyślnie brany jest root. Stąd chyba wzięło się, że

`su` to *superuser*. Przy pomocy `su` możemy przełączać się zachowując ustawienia własnego shella np.:

```
su john
```

albo z ustawieniami danego użytkownika:

```
su - john
```

Po tym zostaniemy zapytani o hasło użytkownika `john`. O hasło nie będziemy pytani, gdy jesteśmy superuserem i używamy `su`, aby przełączyć się na innego użytkownika.

Możemy także użyć `su` aby wykonać tylko jakiś konkretny program z przywilejami danego użytkownika:

```
su root -c "rm /var/log/syslog.7"
```

Tutaj przełączmy się na `root`'a po to tylko by usunąć z systemu jeden plik, czego nie możemy wykonać jako zwykły użytkownik.

Warto wspomnieć przy okazji program `sudo`. Jeśli administrator, czyli superuser, odpowiednio skonfigurował program `sudo`, to podając swoje hasło możemy wykonać dowolny program z uprawnieniami `root`'a. W szczególności tym dowolnym programem może być `su`. Powstaje często używane polecenie:

```
sudo su
```

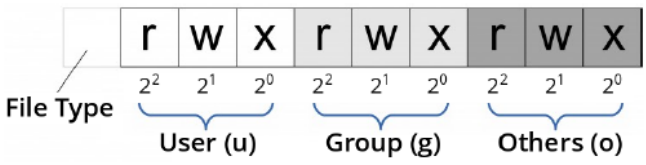
mylnie traktowane jako jeden program. To są dwa programy: pierwszy przełącza nas na chwilę na superusera, drugi przełącza na stałe na superusera. Dla przypomnienia, bo nazwę `root` można pominąć, ten sam efekt da:

```
sudo su root
```

Wróćmy teraz do programu `ls` i tego co pokazuje zawołany z opcją `-l`, na przykład:

```
sirius$ ls -lh
total 967605
drwxr-xr-x 11 mariusz staff       11 Aug 5 2010 archive
drwx------  2 mariusz staff       89 Aug 3 2017 bin
drwxr-xr-x 12 mariusz staff       12 Dec 21 2014 documents
drwxr-xr-x 17 mariusz staff       17 Dec 14 23:42 download
drwxr-xr-x 54 mariusz staff       54 Jun 4 2016 guides
-rw-r--r--  1 mariusz staff       6.3M Jan 2 20:38 httpd-2.4.29.tar.bz2
drwxr-xr-x 45 mariusz staff       45 Aug 18 2017 projects
drwxr-xr-x  2 mariusz staff       46 Dec 28 23:32 public_html
drwx------  9 mariusz staff       9 Feb 20 2008 tex
```

Teraz myślę, że jasne jest znaczenie kolumny 3 i 4, czyli odpowiednio: użytkownik i grupa, którzy są właścicielami pliku (bo katalog to też plik, przekonamy się gdy będzie mowa o UFS). Spróbujemy teraz rozszyfrować kolumnę 1, czyli prawa dostępu do pliku. Jak to zostało powiedziane na samym początku, każdy plik, a więc wszystko w Unixie, ma określone prawa dostępu. 10 znaków w kolumnie 1 można podzielić na 4 składowe: typ pliku (file type), dostęp użytkownika (user), dostęp grupy (group), dostęp pozostałych (other) .



Dostęp	Symbol	Wartość
odczyt	r	4
zapis	w	2
wykonanie	x	1
brak	-	0

Symbol	Typ pliku
-	zwykły plik
d	katalog
l	dowiązanie symboliczne
b	specjalny plik blokowy
c	specjalny plik znakowy
p	potok FIFO
s	gniazdo (socket)
D	door
P	port zdarzenia

Typ pliku to zwykle:

- `-` : zwykły plik,
- `d` : katalog,
- `l` : dowiązanie symboliczne, itd.

Pozostałe 9 znaków dzielimy na 3 zestawy: user, group i other. W każdym z nich znajdują się po 3 znaki:

- `r` : prawo do odczytu,
- `w` : prawo do zapisu,
- `x` : dla zwykłych plików prawo wykonania programu (execute), dla katalogów prawo przejścia do katalogu,
- `-` : brak uprawnień.

Znowu, komputer woli liczby i dlatego powyższym uprawnieniom przypisano odpowiednie wartości numeryczne: 4, 2, 1, 0. Zwróćmy uwagę, że są to kolejne potęgi liczby 2, czyli każdy znak odpowiada jednemu z 9 bitów w metadanych, gdzie zapisane są prawa dostępu do danego pliku. Inaczej mówiąc `rwx` to to samo co 7, `rw-` to 6, `r--` to 4, `--x` to 1 itd. Dlaczego 9 bitów po 3, albo 3 zestawy symboli po 3? Pierwszy zestaw (user) oznacza, co z danym plikiem może zrobić jego właściciel, to znaczy użytkownik o tej samej nazwie jaką widzimy w kolumnie 3 wypisanej przez `ls -l`. Drugi zestaw (group) oznacza co może zrobić grupa, a trzeci zestaw (other) oznacz co mogą z danym plikiem zrobić pozostali, czyli ci, którzy nie są użytkownikiem z kolumny 3 i nie należą do grupy z kolumny 4. Na przykład: `rwxr-x---` dla jakiegoś katalogu oznacza, że właściciel może wszystko, grupa nie ma tam prawa zapisu, ale może odczytywać i zaglądać do tego katalogu, natomiast pozostali nie mają żadnych praw. To samo można wyrazić numerycznie jako 750. Proste prawda?

Może powstać pytanie czy `rwx` to to samo co `wxr`? Dla nas może i tak, ale dla Unixa to drugie nie ma sensu. Istotna jest pozycja, czyli konkretny z 9 bitów. Jeśli ten bit jest 0 to -, natomiast gdy ma on wartość 1 to bity 3, 6, i 9 będą reprezentowane jako `r`, bity 2, 5 i 8 będą reprezentowane jako `w`, a bity 1, 4 i 7 będą reprezentowane jako `x`. Człowiek szybciej odczyta uprawnienia z wersji symbolicznej, niż z ciągu bitów. Na przykład `110100100` oznacza `rw-r--r--`. Co łatwiej odczytać?

System bardzo konsekwentnie przestrzega uprawnień z uwzględnieniem tego, kto należy do jakiej grupy. Powiedzmy że mamy plik:

```
-rw----rw-   1 root   staff      22 lis 15 20:16 a.txt
```

Co może z nim zrobić użytkownik `mariusz`, który należy do grupy `staff`? Nic. Użytkownik `mariusz` nie jest właścicielem tego pliku, natomiast grupa `staff` nie ma żadnych uprawnień. W tym

wypadku `mariusz` nie kwalifikuje się jako pozostali (other) bo spełnia warunek członkostwa w grupie, która jest właścicielem powyższego pliku. Administratorzy Unixa czasem mylnie uważają, że uprawniania pozostałych (other) przechodzą również na właściciela (user) albo grupę (group).