Progresivní technologie v informatice II

Operační systémy

Pokročilá správa, administrátor, ACL, privilegia, pokročilé filesystémy

Fakulta informačních technologií
České vysoké učení technické v Praze

Financováno Evropskou unií
NextGenerationEU
Národní plán obnovy
MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

Unix Administrator

- ***Conventional*** name in Unix-like systems - **root**
- In fact defined by **UID 0** (formerly) and/or by **full set of privileges** (in current ”secure” systems)
- Has absolute access to the system
- Examples of root activities:
  - System shutdown, restart, run level change
  - Starting and stopping system services (daemons)
  - Software installation, maintenance and patching the system
  - Kernel configuration
  - Working with devices (special files)
  - Unlimited access to all files and commands
  - Administration of filesystems, users, network interfaces….
  - Setting system limits and priorities
  - Opening system (0..1023) ports

Nabytí práva administrátora

3 methods:

- Log-in as a root
- **su** *command* (**s**witch **u**ser)
- **sudo** *command* (**su**per user **do**)

Authentication – verifying "who are you"

methods: password, keys, fingerprint,… ⇨ PAM

Authorization – verifying "are you permitted to do something"
(previous authentication is expected)

methods: identity, group membership, RBAC,….

Login

"Standard" way – direct log-in as a root
(now is taken as a non secure)

Problems, differences and restrictions:
console, local login, remote login
password, ssh keys

Restriction configuration files:

`/etc/ssh/sshd_config`

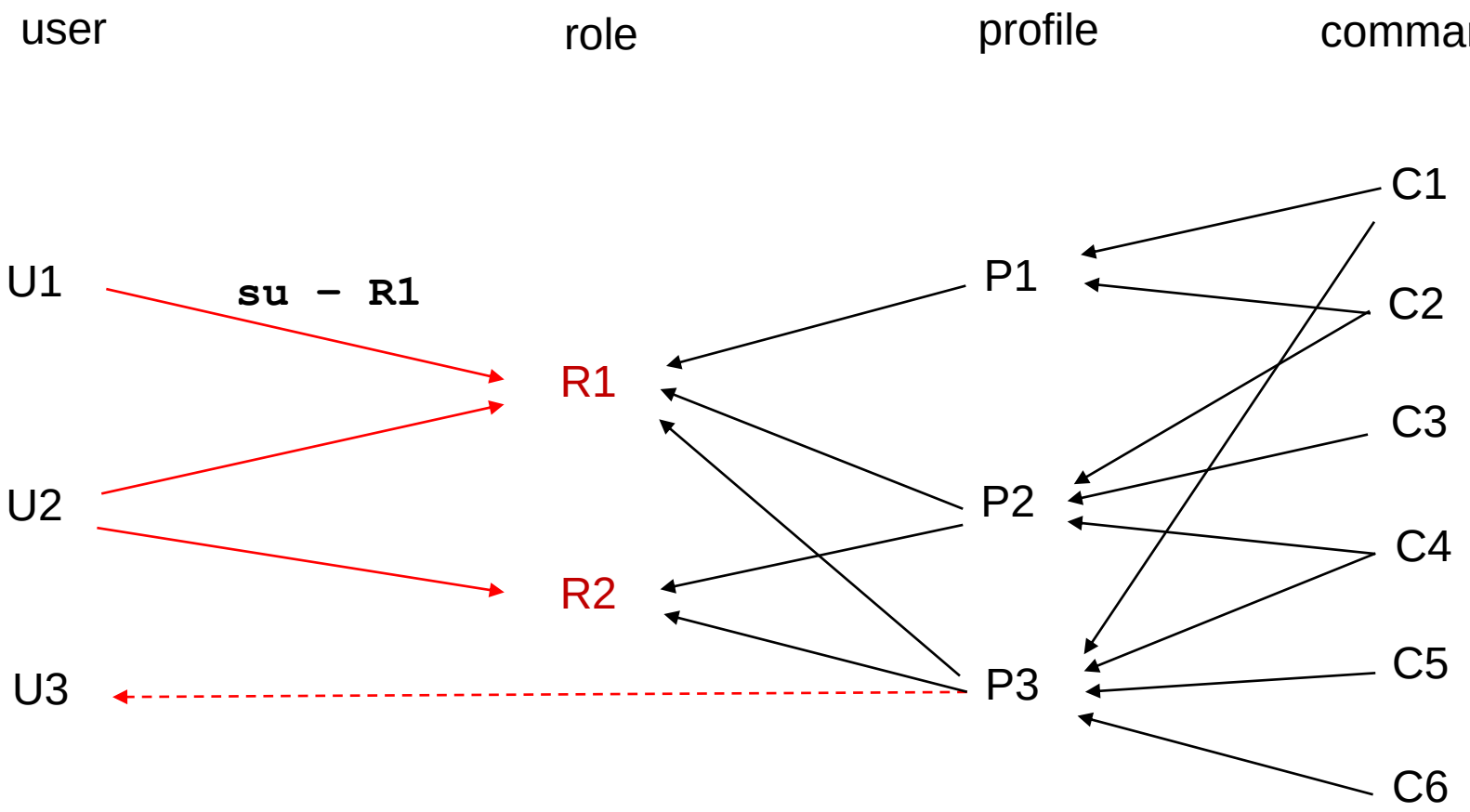
`/etc/default/login` (Solaris)

`/etc/security/pwquality.conf` (Linux)

How to solve security problem of more administrators?

Correct solution: **roles** !

RBAC - diagram



Role (with prof.) assigned to user and he becomes it by **su** correct

Profile is assigned directly to the user not recommended

Privilegia

Traditional UNIX system

- Authorization to execute a privileged (mount, read,…) operation is defined by UID of the process
- Kernel tests if process UID is 0
- ⇨ Administrator's authorization (competency) is individable

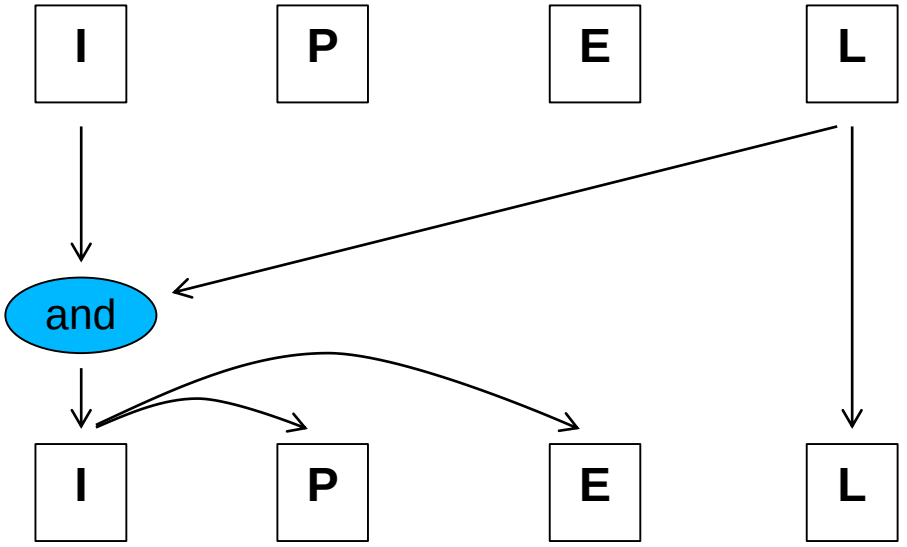
System supporting privileges

- Approximately 80 privileges covering all operations are defined
- UID does not affect, process carries a set of privileges with it
- Default setting:
  - root process has **all** privileges assigned
  - „Common" user process has a **subset** of privileges allowing to perform „common" (non root) operations only
- From the implementation point of view, privileges are in 4 sets:
  (**E**: effective, **I**: inheritable, **P**: permitted, **L**: limit)
- Kernel tests privilege to execute the specific operation from the set **E**

Nastavování privilegií

- Default privileges in Solaris are set in the file:
  `/etc/security/policy.conf`
- Can be set (assigned) within RBAC
- Can be modified dynamically

- Principle of inheritance to the child process:



Příkaz su

`/bin/su [-] [username [argument]]`

- See the Lecture 1
- Changing all 3 identities of the user to the identity ***`username`***
- Can be used for single execution of command under different identity

Rules:

- **root** can change identity without user's password
- Missing username: as if **root** username was used
- Missing `-` : identity is changed, environment remains - **dangerous!**
- Used `-` : equivalent (almost) of new login

Examples

```
su username    ; id ; pwd
su - username ; id ; pwd
exit; id
```

Příkaz sudo

`/bin/sudo [-u username]` ***`command arguments`***

- Allows user to run *command* with identity/privileges of another user (typically root), defined by **–u** option
- If **–u** option is omitted, it is the same as **–u root**
- Controlled by configuration file: **/etc/sudoers**

Benefits and risks

- Delegation of only specific commands to specific users (+)
- Can be monitored by **syslog** tool (+)
- Configuration file can be shared within more systems (+)
- Errors in configuration file (–)
- Misusing of non-privileged account can cause “privileged” destruction (–)

Konfigurace sudo

`/etc/sudoers` file

- Can be edited by root only
- Direct editing using any text editor
- Using **`visudo`** command – syntax checking
  (editor is optional, defined by **EDITOR** variable)
- In case of syntax error it does not work
- “Standard“ syntax of sudoers line:

  who where=(as who) what

  Note: If **(as who)** field is omitted it the same as **(root)**

Examples:

```
user1  host1=(root)  /usr/sbin/shutdown
user3 ALL=(user4,user5) /bin/kill,/bin/pkill
# aliases (like ALL above) can be defined
```

ACL – Access Control List

```
-rwxrw-r--+ 1  george student ..... filename
```

- “Standard” file-permissions (3 user categories) were not enough
- ACL – extended list of file permissions attached
- Allow to define specific permissions (rwx) for specific users and/or groups
- **mask** can be set and it allows to manipulate all ACL assigned to the file
- Implemented in many operating systems but not in all file systems – incompatible (mostly)
- Typically implemented using one more i-node (shadow i-node) linked from ”standard” one

ACL – příkazy

```
$ setfacl -m user:paul:6 filex

$ getfacl  filex

# file: filex
# owner: george
# group: student
user::rwx
user:paul:rw-                   #effective:rw-
group::---                      #effective:---
mask:rwx
other:--x
```

NFSv4 ACL

- New (extended) implementation
- Contains some new “non traditional Unix“ permissions
- Instead of **setfacl** and **getfacl** extended **chmod** and **ls –v** commands

```
chmod A[index][+-=]ACLspec filename
```

chmod A+user:john1:rwx:allow nfs.dir
chmod A-user:john2:addfile:allow nfs.dir
chmod A+user:john3:x:deny nfs.dir

ls -v nfs.dir

Problémy stávajících filesystémů

- Vznik nekonzistencí při nekorektním vypnutí
- Nemožnost (nebo komplikované) zvětšování fyzické kapacity filesystému
- Techniky RAID (sw řešení), snapshot, zálohování atd. jsou řešeny mimo vlastní filesystém a jsou s ním pouze více či méně kompatibilní
- Filesystémy nejsou hierarchické
- Komplikovaná administrace
  - mnoho různých příkazů, konfiguračních souborů, …
  - konfigurace často mimo filesystém (soubory v /etc) – problém při přenosu
- Chybějící vlastnosti:
  např. klonování, cache, kryptování, deduplikace….

Princip ZFS (Zettabyte File System)

- Na úrovni metadat (atributy souboru, i-node tabulka, linky…) stejné jako UFS, implementováno ale jinak.
- Datový prostor tvoří virtuální datová oblast – **pool**
- **Pool** je tvořen zdroji dat: **disky**, **partitions** (**slices)** a **soubory** (virtuální zařízení - device)
- Z poolu se alokují **datové bloky** – různá velikost, tvořené nad strukturami RAID
- Nad nimi “vyhledávací“ strom, tvořen bloky z téhož poolu
- Každý blok má kontrolní součet (hash, 256 bitů), který je mimo vlastní data, v jiném (nadřazeném) datovém bloku.
  Obsah datových bloků může být kryptovaný a/nebo komprimovaný.

Princip ZFS (Zettabyte File System)

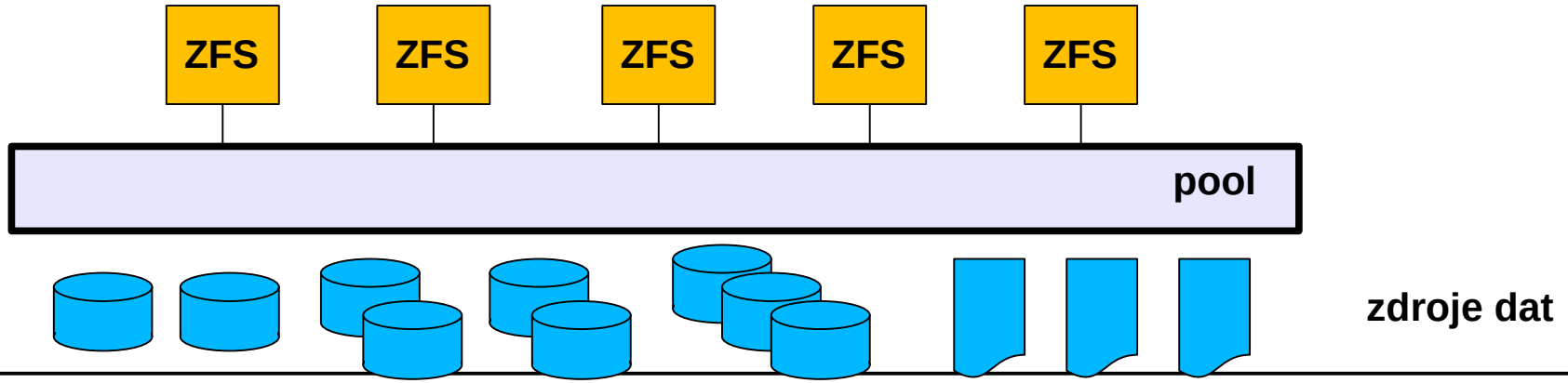
- Transakční systém - princip „**copy on write**“

  Data se nikdy nepřepisují, bloky se zkopírují, změní a transakce je pak jako celek přijata nebo odmítnuta → jakkoli složitá operace je pak tzv. atomická → vždy konzistentní

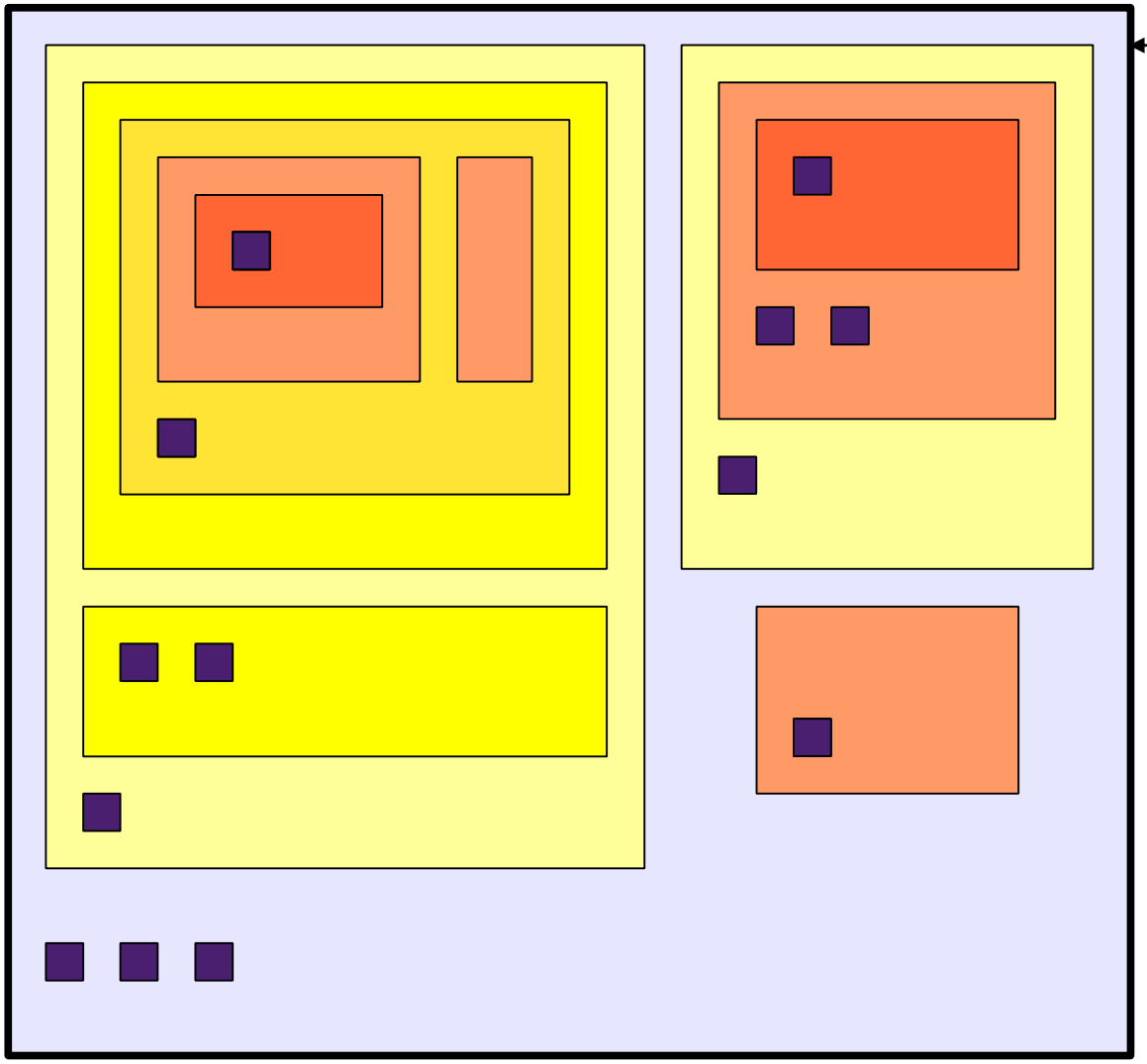
- **Pool** sám je (zfs)filesystém, v něm je možno hierarchicky vytvářet další

- Vlastnosti (atributy, properties) zfs filesystému jsou jeho součástí a nese si je s sebou.
  Každá má “**default**“ hodnotu a je **ro/rw** a **dedičná/nedědičná**.

  Příklady: verze ZFS (ro, dědičná), mountpoint (rw, nedědičná)



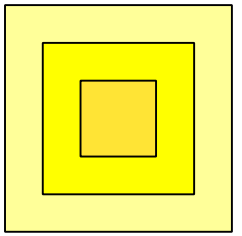
Hierarchie ZFS



pool

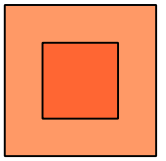
```
zpool create
zpool add
zpool list
```

ZFS filesystémy



```
zfs create
zfs list
```

adresáře

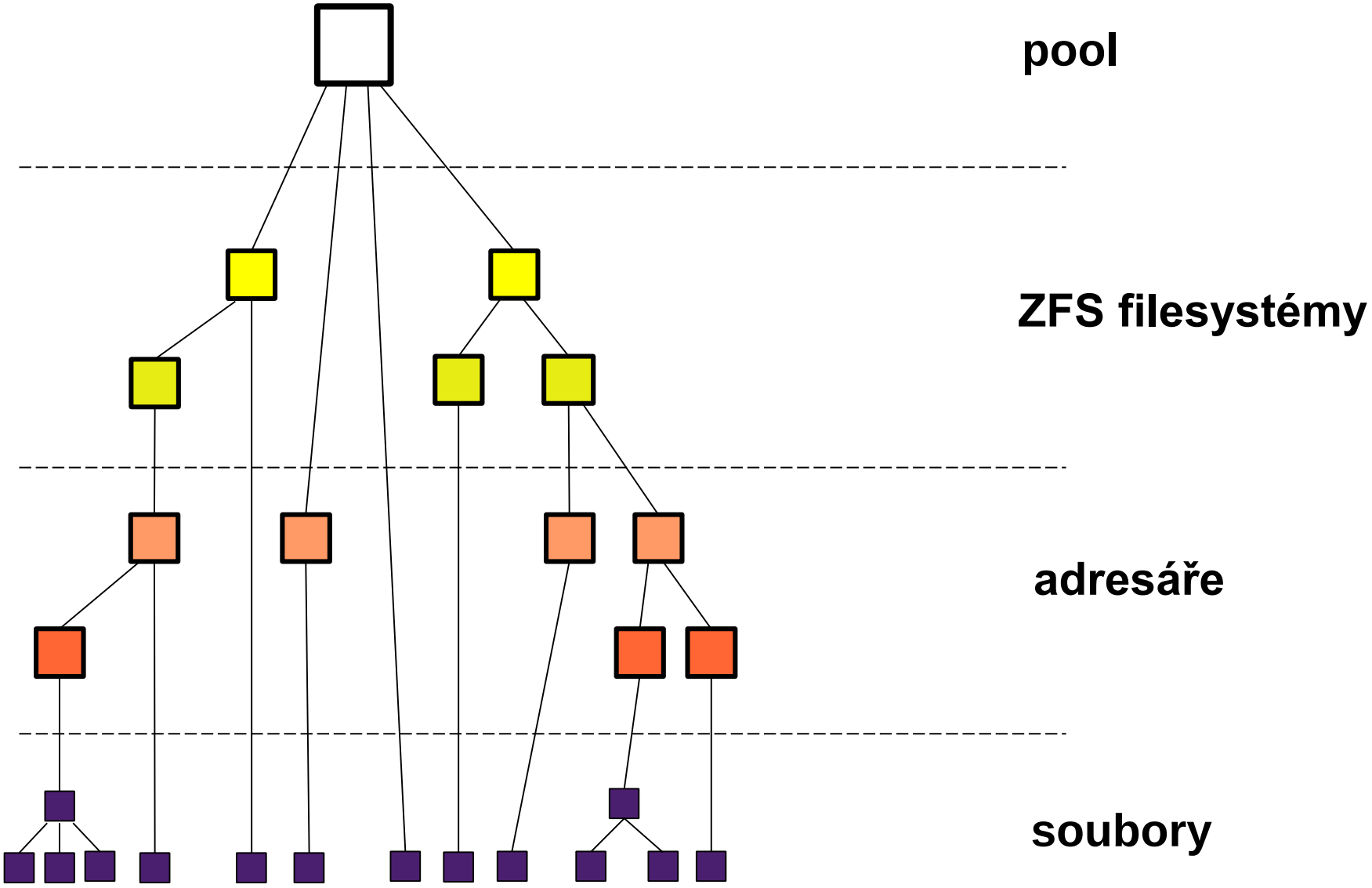


```
mkdir
ls
```

soubory

```
touch
ls
```

Jiný pohled



ZFS

Typy poolů

- standalone (non-redundant), ”striped” princip
- mirrored
- RAID-Z (raidz/raidz1, raidz2, raidz3)

Urychlení:

- pool + **cache** zařízení — urychlení čtení
- pool + **log** zařízení (ZIL – ZFS Intent Log) — urychlení zápisu

  nejlépe použít ssd disky, ZIL – zrcadlení!

Bezpečnost:

- pool + hot spare

Nejen **detekce,** ale i **oprava** chyb (self-healing)
Jak je to možné?

- kontrolní součty mimo blok
- redundantní typ poolu (mirror, raidz, …..)

ZFS - RAIDZ

RAIDZ (RAID2Z, RAID3Z)

- Eliminují tzv. ”**write hole**”
- Co to je „RAID 5 write hole“?
- Jak vzniká a proč „přežívá“?
- Proč v RAIDZ není?
  - Princip “copy on write“ způsobuje, že zápis je atomický, tj. každý zápis je “full stripe write“

- Dále: proměnná délka bloku → dynamická velikost (šířka) stripe → úspora místa, zrychlení
  - Možné díky spojení filesystému a volume manageru

ZFS - příkazy

Pouze 2 příkazy (+ „podpříkazy“)

- **`zpool`** (volume manager)
- **`zfs`** (file system)

Příklady:

```
zpool create .....
zpool list
zpool status

zfs create ...
zfs get ...
zfs set ...
zfs list
```

atd. ...

ZFS

Vytvoření poolu (1)
Přidání device do poolu (2) - okamžitě k dispozici pro všechny datasety
Připojení další komponenty do mirror (3)

```
zpool create tank c0t0d0p0                              (1)
zpool add tank c1t0d0p1                                    (2)
zpool attach mirroredtank c1t1d0p0 c1t0d0p0    (3)
```

Další (sub)příkazy:

```
  detach, split, export, import, offline, online, replace, spare,
  ...
```

```
man zpool
man zfs
```

ZFS

Dataset

jedna z entit: filesystem, snapshot, clone

Properties

```
zpool get all tank                  # vypis vsech
zfs get all tank/home             # vypis vsech
```

read-only properties: nejsou dědičné
settable (nastavitelné): většinou dědičné

Dědičnost

Vyplývá z hierarchie zfs filesystémů a jejich vlastností

Info

```
zpool list
zpool status
zpool iostat [-v]
zpool history
```

ZFS – příklady příkazů

```
zpool create tank mirror c0t0d0p0 c1t0d0p0
zpool add tank mirror /file1 /file2       # nedoporuceno
zpool add tank c2t1d0p0                   # nedoporuceno
zpool export tank

zfs create tank/home
zfs create tank/home/stud
zfs get all tank/home       # vypis vlastnosti tank/home
zfs set mountpoint=/export/home tank/home
zfs set mountpoint=legacy tank/home/stud
zfs set quota=100G tank/home/stud
zfs set sharenfs=on tank/home/stud

zfs destroy tank/home
zpool history
zpool destroy tank
```

ZFS – příklady příkazů

```
zpool create tank mirror c0t0d0p0 c1t0d0p0
zpool add tank mirror /file1 /file2        # nedoporuceno
zpool add tank c2t1d0p0                    # nedoporuceno
zpool export tank

zfs create tank/home
zfs create tank/home/stud
zfs get all tank/home        # vypis vlastnosti tank/home
zfs set mountpoint=/export/home tank/home
zfs set mountpoint=legacy tank/home/stud
zfs set quota=100G tank/home/stud
zfs set sharenfs=on tank/home/stud

zfs destroy tank/home
zpool history
zpool destroy tank
```

komprese, kryptování, deduplikace

komprese

komprimují se bloky

algoritmus volitelný, default LZJB (Jeff Bonwick, Sun)

deduplikace

na úrovni poolu, deduplikují se bloky nebo soubory

tabulka otisků

- SHA256 (default) nebo lze zvolit
- lze důvěřovat nebo verifikovat

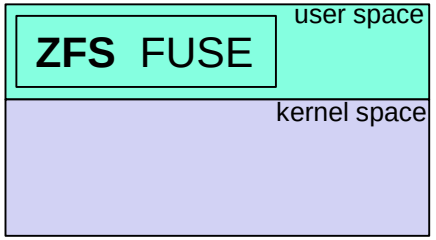
```
zfs set compression=on  tank/data    # LZJB
zfs get compressratio  tank/data
zfs set compression=gzip  tank/data # default gzip-6
                                    # mozne urovne 0-9
zfs set dedup=on  tank/pictures
zfs set dedup=verify  pool/creditcards
zpool get dedupratio  pool
```

ZFS a Linux

Není a nebude – nekompatibilní licence CDDL (ZFS) a GPL (Linux)

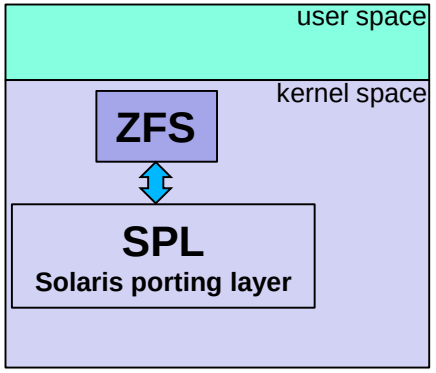
Existují pouze různá (a různě špatná) náhradní řešení:

1.



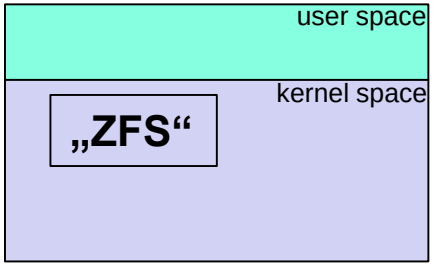
pomalé

2.



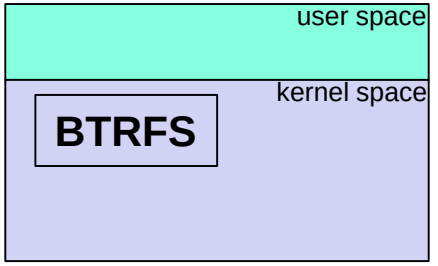
rychlejší,
používané

3.



nová implementace,
rychlejší než 2.,
zpoždění za „originálním“ ZFS,
problémy

4.



jiná (ZFS like) implementace,
jinak totéž jako 3.

Ve všech případech jde pouze o základní funkce (ZFS)filesystému bez přidané funkcionality (např. integrované NFS nebo CIFS)