Transparentní restartování služeb v Linuxu

Knihovna libzedr

Václav Šraier • vaclav.sraier@nic.cz • 09.09.2020

Aktualizace vedou k výpadku služeb

Downtime způsobuje problémy...

Downtime způsobuje problémy...

Tak se mu pojďme vyhnout!

Čeho chceme dosáhnout?

aktualizovat kód běžící služby

bez významných viditelných následků

File descriptory

- reference na kernelový objekt (typicky IO)

- při volání syscallu exec() zůstávají zachovány
- při volání syscallu fork() se kopírují
- dají se poslat skrz Unix socket jinému procesu

Jak to dělá systemd?

- jednovláknový proces
- `systemctl daemon-reexec`
- průběh restartu
  1. serializace stavu
  2. `exec` nové binárky systemd
  3. deserializace stavu

Jak se restartuje Knot Resolver?

- více jednovláknových procesů
- procesy poslouchají na stejném socketu
- průběh restartu
  1. spustí se nový proces přímo z binárky
  2. starý proces se vypne (graceful shutdown)
  3. opakuje se pro všechny staré procesy

Jak to dělá nginx?

- více-procesová architektura
  - master process a worker procesy
- průběh restartu
  - `SIGUSR2` způsobí, že master spustí nový master
  - `SIGWINCH` na starý master proces ukončí workery
  - requesty zpracovává již pouze nová instance

Všechno jsou to ale velké projekty...

Všechno jsou to ale velké projekty...

Zkusme to zobecnit a zjednodušit!

Demo

libzedr

- **lib**rary for **ze**ro-**d**owntime **r**estarts
- zdrojáky: https://gitlab.com/vakabus/libzedr
- cíl
  - jednoduchost na použití v nových i existujících projektech
  - umožnit kooperaci se správci služeb

- vyvíjeno mnou – Vašek Šraier • **vaclav.sraier@nic.cz**
  pod dohledem – Michal Koutný • **mkoutny@suse.com**

Příklad služby

```c
#include <libzedr.h>

int init_fd(void* arg);

int main(int argc, char** argv) {
    zedr_init(argc, argv);
    /* ... */
    int fd = zedr_fd_init(init_fd, NULL);
    /* ... */
    while (true) {
        int nfd = zedr_accept(fd, &addr,sizeof(addr));
        /* ... */
    }
}
```

libzedr API

- `void zedr_init(int argc, char** argv);`
- `int zedr_fd_init(fd_init func, void* arg);`
- `void zedr_restart_point(void);`

libzedr API

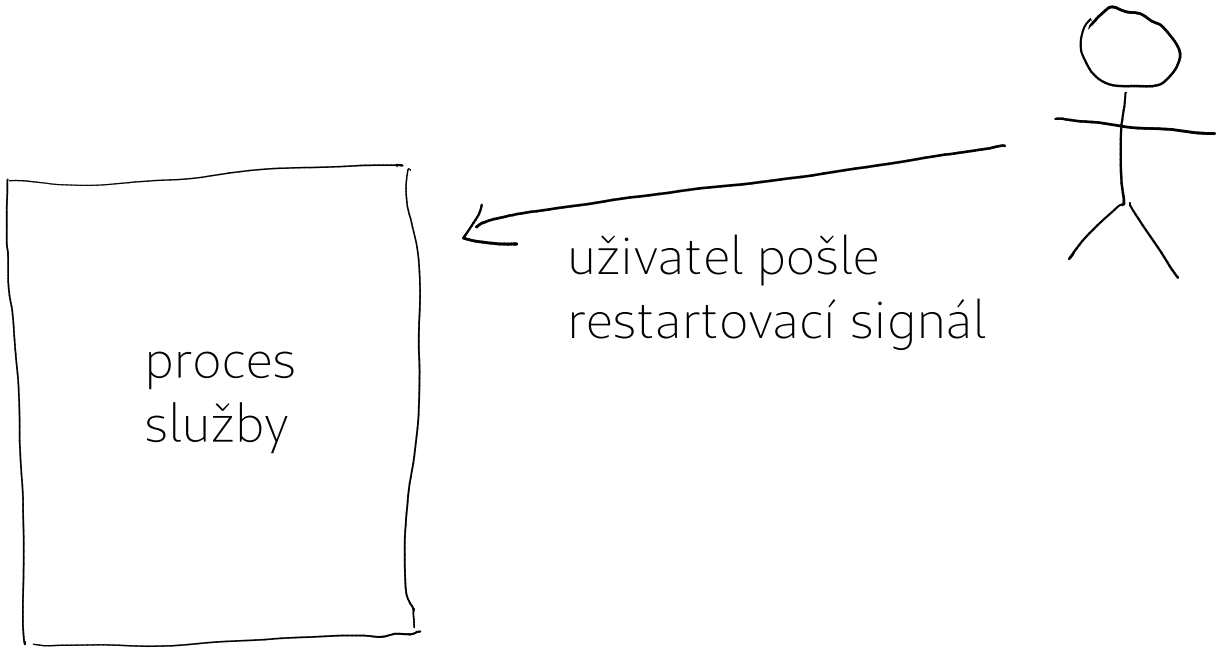
- `void zedr_init(int argc, char** argv);`
- `int zedr_fd_init(fd_init func, void* arg);`
- `void zedr_restart_point(void);`

- `void zedr_config_*(...)`

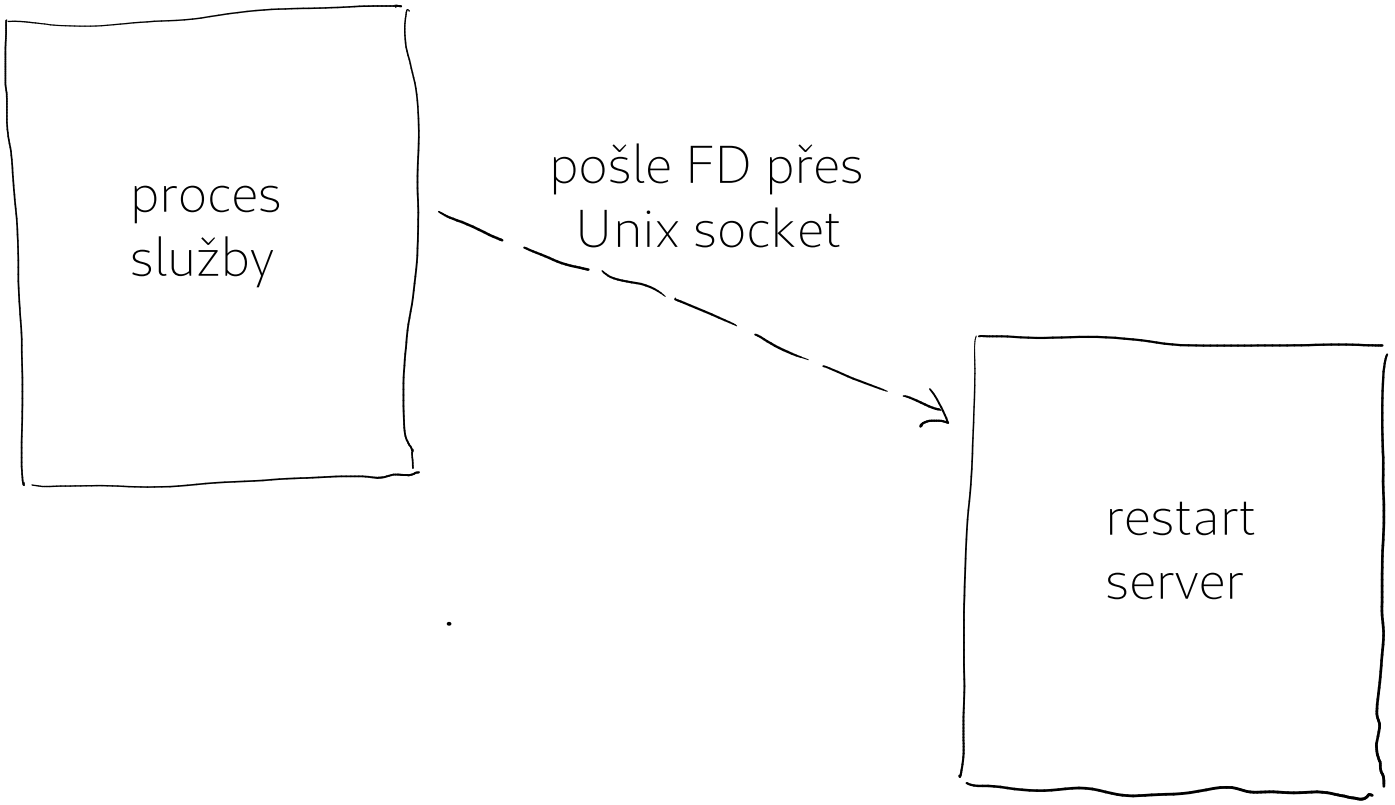
- `int zedr_poll(...)`
- `int zedr_accept(...)`
- `int zedr_recv(...)`

Jak libzedr funguje?

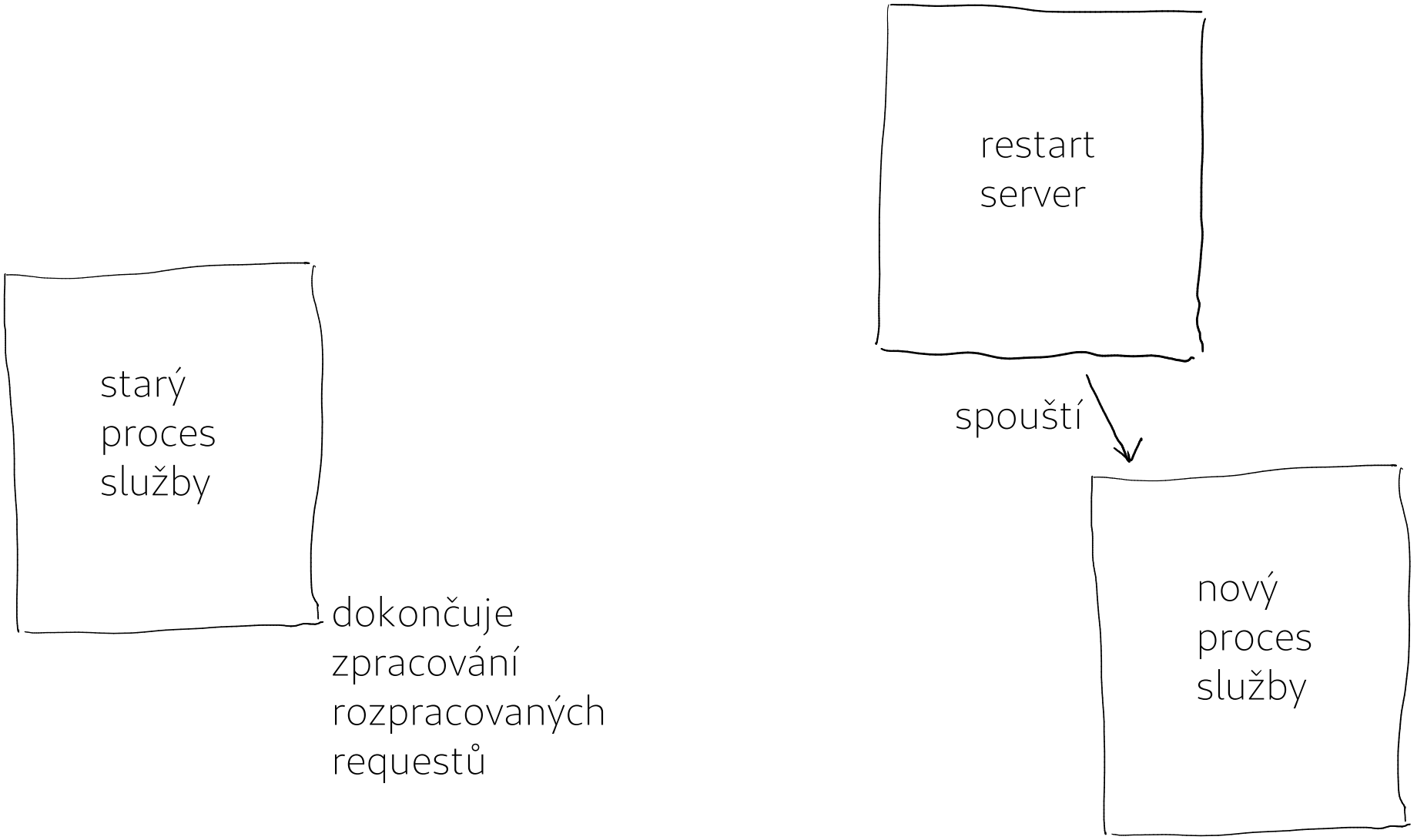
Jak libzedr funguje?



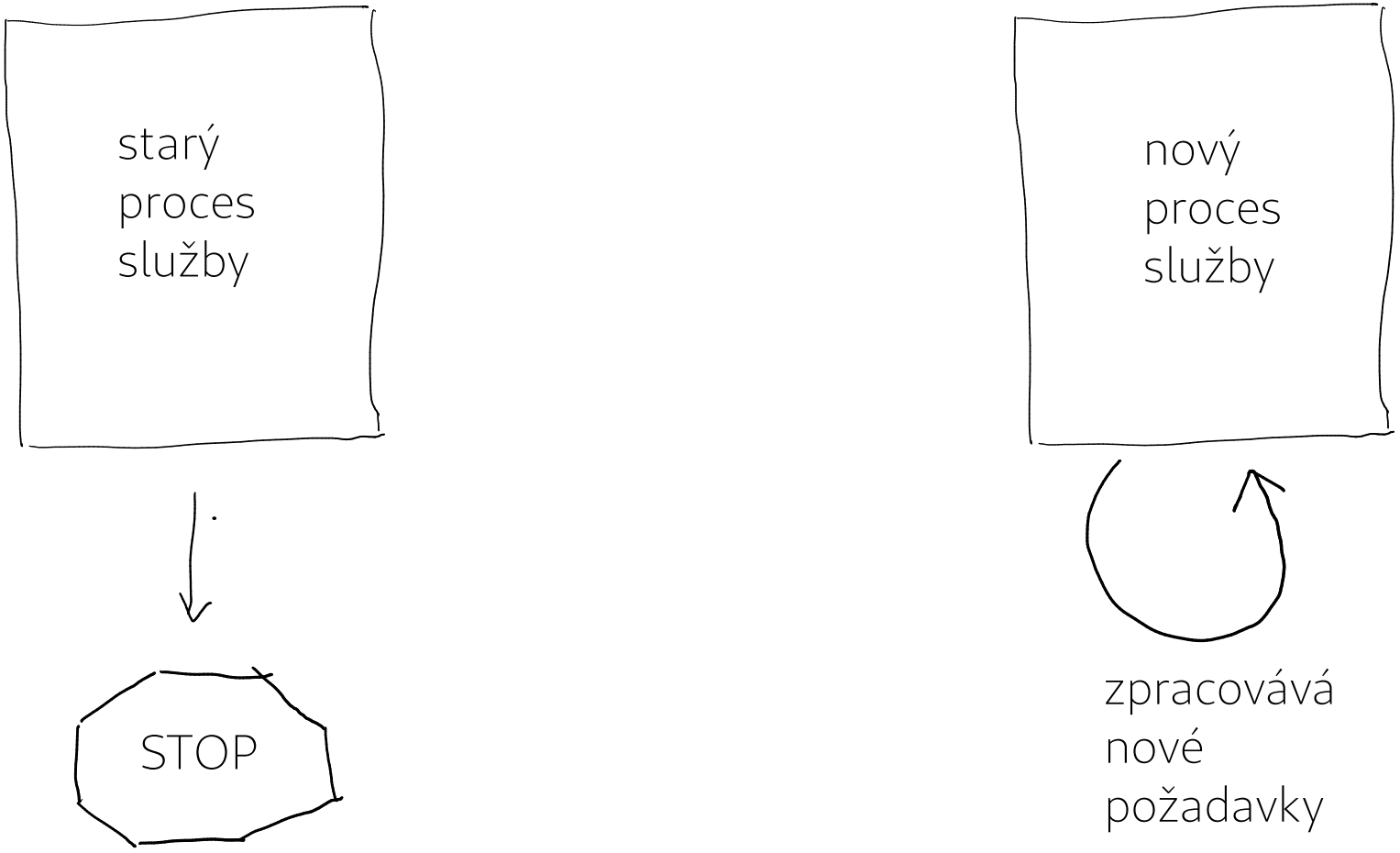
Jak libzedr funguje?



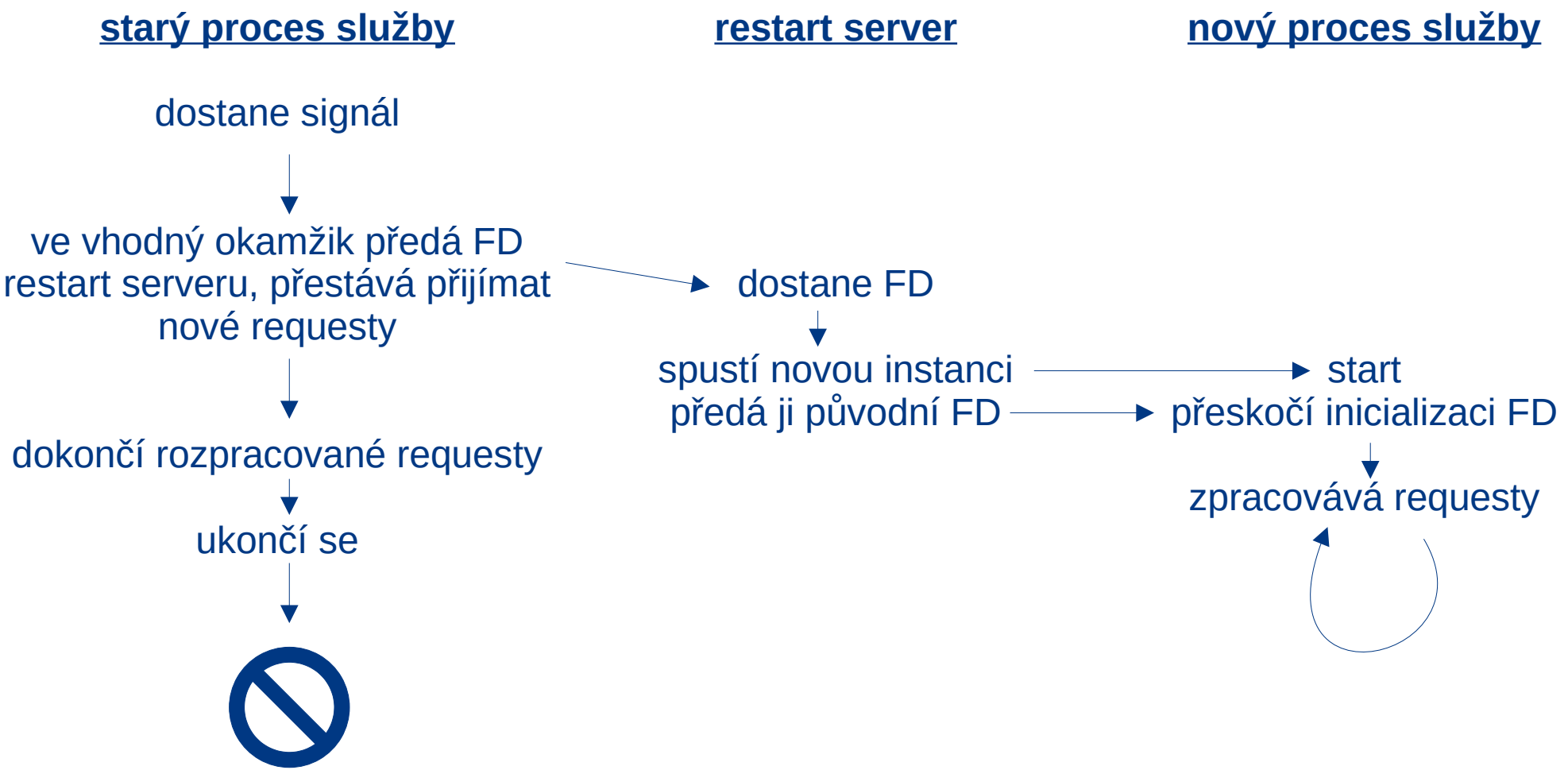
Jak libzedr funguje?



Jak libzedr funguje?



Shrnutí


starý proces služby
restart server
nový proces služby
dostane signál
ve vhodný okamžik předá FD restart serveru, přestává přijímat nové requesty
dokončí rozpracované requesty
ukončí se
dostane FD
spustí novou instanci
předá ji původní FD
start
přeskočí inicializaci FD
zpracovává requesty

Příklad služby

```c
#include <libzedr.h>

int init_fd(void* arg);

int main(int argc, char** argv) {
    zedr_init(argc, argv);
    /* ... */
    int fd = zedr_fd_init(init_fd, NULL);
    /* ... */
    while (true) {
        int nfd = zedr_accept(fd, &addr,sizeof(addr));
        /* ... */
    }
}
```

Restart server

- vyměnitelný
  - může to být správce služeb
  - integrovaná implementace uvnitř knihovny pro restart bez externí pomoci

- experimentální integrace do systemd
  - https://gitlab.com/vakabus/libzedr_systemd

libzedr

- knihovna na transparentní restarty
- s minimalistickým API
- podpora integrace do správců služeb

Vývoj do budoucna

- libzedr API je stabilní
- integrace do systemd
  - sledování různých generací služby
  - předělat backend knihovny tak, aby využívala existující systemd API
  - upstream?

Děkuji za pozornost!
Dotazy?

Václav Šraier • vaclav.sraier@nic.cz