

Operacijski sustavi (OS)

Nositelj: doc. dr. sc. Ivan Lorencin

Asistent: Luka Blašković, mag. inf.

Ustanova: Sveučilište Jurja Dobrile u Puli, Fakultet informatike u Puli



Fakultet informatike u Puli

(5) Rad na Virtualnom stroju: Naprednije teme

#5

OS

Procesi predstavljaju aktivne instance programa koji se izvršavaju unutar operacijskog sustava. U više zadatačnim operacijskim sustavima, što uključuje sve suvremene računalne sustave, procesor se brzo prebacuje između više procesa, omogućujući tako istovremeno i konkurentno izvođenje različitih programa. Operacijski sustav ima ključnu ulogu u organizaciji i optimizaciji rada sustava – on upravlja dostupnim resursima te raspodjeljuje procesorsko vrijeme među različitim procesima i njihovim dretvama. U ovoj skripti pružit ćemo uvod u upravljanje procesima i dretvama, te upoznati osnovne alate koji se koriste za njihovo praćenje i kontrolu u Linux operacijskom sustavu. Osim toga, obraditi ćemo i upravljanje korisnicima, korisničkim grupama i servisima – pozadinskim procesima koji se automatski pokreću prilikom dizanja sustava i omogućuju kontinuiran rad ključnih funkcionalnosti operacijskog sustava.

Posljednje ažurirano: 21.5.2025.

- gradivo skripte je dovršeno, zadaci na kraju će se dodati naknadno - uskoro

Sadržaj

- [Operacijski sustavi \(OS\)](#)
- [\(5\) Rad na Virtualnom stroju: Naprednije teme](#)
 - [Sadržaj](#)
- [1. Upravljanje procesima](#)
 - [1.1 Interaktivni pregled procesa \(`top` / `htop`\)](#)
 - [Alat `top`](#)
 - [Alat `htop`](#)
 - [Zadatak 1: Tumačenje procesa](#)

- [1.2 Programske dretve](#)
 - [Alat kill](#)
 - [Alati pidof i pgrep](#)
 - [Alat pkill](#)
- [1.3 Alat nice](#)
- [Zadatak 2: Upravljanje procesima](#)
- [2. Upravljanje korisnicima](#)
 - [2.1 Naredba useradd](#)
 - [2.2 Naredba usermod](#)
 - [2.3 Naredba userdel](#)
 - [Zadatak 3: Upravljanje korisnicima i grupama](#)
- [3. Upravljanje servisima](#)
 - [Zadatak 4: Upravljanje servisima](#)
- [Zadaci za Vježbu 5](#)

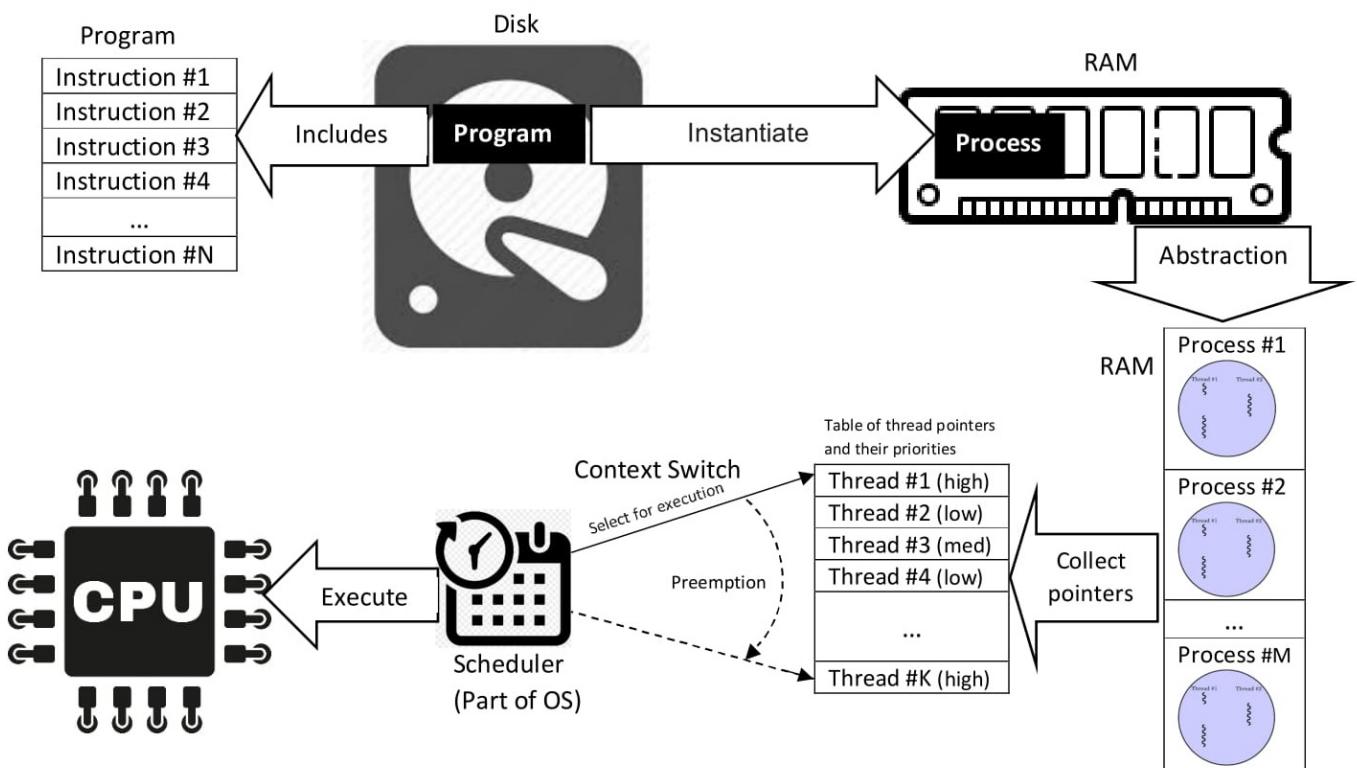
1. Upravljanje procesima

Puno puta smo čuli pojma proces, no što je to zapravo? Najjednostavnije rečeno, **proces** je instanca programa koja se izvršava.

Pojam **računalnog programa** možemo vrlo slično definirati - kao skup instrukcija koje se izvršavaju u određenom redoslijedu, a tipično se nalaze pohranjene na disku. Jednom kad se program pokrene, te instrukcije se učitavaju u radnu memoriju i izvršavaju. U tom trenutku, program postaje proces. Međutim, često je slučaj da se više procesa pokreće istovremeno, odnosno da se **više procesa asocira s istim programom**.

Osim samog programa, proces sadrži (enkapsulira) i sve druge resurse potrebne za njegovo izvršavanje, kao što su:

- **adresni prostor** (eng. *address space*) - dio memorije koji proces koristi
- **skup otvorenih datoteka** (eng. *open files*) - datoteke koje proces koristi
- **deskriptori procesa** (eng. *process descriptors*) - informacije o procesu, kao što su PID (eng. *process ID*), UID (eng. *user ID*), GID (eng. *group ID*) i drugi atributi
- **varijable okoline** (eng. *environment variables*) - varijable koje proces koristi
- **informacije o procesoru** (eng. *CPU*) - trenutno stanje procesora (registri, brojač instrukcija, itd.) ako je proces u stanju izvođenja



Računalni program je skup instrukcija pohranjen na disku, dok proces predstavlja egzekuciju (realizaciju) tog programa. Naravno, za realizaciju je potrebno više od samih instrukcija (programskog koda)

Višezadaćnost (eng. *multitasking*) je metoda koja dozvoljava izvođenje više procesa istovremeno - konkurentno (eng. *concurrent*). U višezadaćnom operacijskom sustavu, novi procesi mogu biti stvorenici i izvršavati se paralelno s postojećim procesima.

Svaka jezgra procesora (eng. *CPU core*) može izvršavati samo jedan proces u jednom trenutku. Međutim, višezadaćnost dozvoljava procesoru da se **brzo prebacuje** (eng. *switch*) između različitih procesa bez čekanja da jedan proces završi.

Mehanizam kojim se ostvaruje ovo prebacivanje između procesa naziva se *context switching*. Termin se koristi i u psihologiji, gdje se, baš kao i u računalstvu, koristi za opisivanje prebacivanja između različitih zadataka, projekata ili aktivnosti. Premda prema nekim istraživanjima, "biološki" *context switching* [često dovodi do smanjenja produktivnosti](#), što definitivno nije slučaj u računalstvu.

Glavna zadaća OS-a za vrijeme višezadaćnosti je učinkovito upravljanje resursima i raspodjela vremena procesora između različitih procesa.

1.1 Interaktivni pregled procesa (`top/htop`)

Podignut ćemo naš virtualni stroj i Ubuntu Server. Kako bi provjerili sve procese koji se trenutno izvršavaju, možemo koristiti naredbu `ps aux`.

```
→ ps aux
```

Dobit ćemo detaljni tabični prikaz svih aktivnih procesa, možemo rezultat proslijediti i kroz naredbu `less` kako bi mogli *proscrollati*:

```
→ ps aux | less
```

Međutim, rekli smo da u višezadaćnom operacijskom sustavu procesor vrlo često prebacuje (skače) između različitih procesa. Samim time, procesi se mogu nalaziti u različitim stanjima. Naredba `ps` ne daje nam mogućnost da pratimo promjene stanja procesa, već daje **snapshot procesa u trenutku kada je naredba izvršena**.

Alat `top`

Kako bi pratili promjene stanja procesa, možemo koristiti ugrađeni alat `top`.

`top` pruža pregled svih aktivnih procesa i njihovih resursa u stvarnom vremenu.

```
→ top
```

```

Ubuntu Server [Running]
top - 21:47:27 up 51 min, 1 user, load average: 0.08, 0.02, 0.01
Tasks: 96 total, 1 running, 95 sleeping, 0 stopped, 0 zombie
%Cpu(s): 3.9 us, 5.1 sy, 0.0 ni, 89.8 id, 1.2 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 1954.4 total, 1455.0 free, 254.4 used, 324.1 buff/cache
MiB Swap: 1919.0 total, 1919.0 free, 0.0 used, 1700.1 avail Mem

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
 1 root 20 0 21852 12012 8556 S 1.0 0.6 0:01.60 systemd
1362 root 20 0 12915 8320 6400 S 1.0 0.4 0:00.03 wget
288 root 19 -1 66680 16072 15048 S 0.7 0.8 0:00.45 systemd-journal
1341 lukabla+ 20 0 9212 5120 3072 R 0.7 0.3 0:00.03 top
589 systemd+ 20 0 21495 12160 10112 S 0.3 0.6 0:00.13 systemd-resolve
738 syslog 20 0 222812 4608 3584 S 0.3 0.2 0:00.09 rsyslogd
1342 root 20 0 2380 1408 1408 S 0.3 0.1 0:00.01 50-motd-news
 2 root 20 0 0 0 0 S 0.0 0.0 0:00.01 kthreadd
 3 root 20 0 0 0 0 S 0.0 0.0 0:00.00 pool_workqueue_release
 4 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker/R-rcu_g
 5 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker/R-rcu_p
 6 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker/R-slub_
 7 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker/R-netns
 8 root 20 0 0 0 0 I 0.0 0.0 0:06.34 kworker/0:0-events
 9 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker/0:0H-kblockd
11 root 20 0 0 0 0 I 0.0 0.0 0:01.32 kworker/u4:0-events_power_efficie+
12 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker/R-mm_pe
13 root 20 0 0 0 0 I 0.0 0.0 0:00.00 rcu_tasks_kthread
14 root 20 0 0 0 0 I 0.0 0.0 0:00.00 rcu_tasks_rude_kthread
15 root 20 0 0 0 0 I 0.0 0.0 0:00.00 rcu_tasks_trace_kthread
16 root 20 0 0 0 0 S 0.0 0.0 0:00.26 ksoftirqd/0
17 root 20 0 0 0 0 I 0.0 0.0 0:00.33 rcu_preempt
18 root rt 0 0 0 0 S 0.0 0.0 0:00.13 migration/0
19 root -51 0 0 0 0 S 0.0 0.0 0:00.00 idle_inject/0
20 root 20 0 0 0 0 S 0.0 0.0 0:00.00 cpuhp/0
21 root 20 0 0 0 0 S 0.0 0.0 0:00.00 cpuhp/1

```

ChatOPF can make mistakes. Check important info. See [About Preferences](#).

Rezultat naredbe `top` je interaktivni prozor koji prikazuje dinamičku listu procesa u realnom vremenu

Na vrhu prozora možemo vidjeti nekoliko informacija o sustavu, uključujući:

- `Tasks` - ukupan broj procesa (*taskova*) i podjela prema stanju
- `%Cpu(s)` - utilizacija procesora podijeljena po više kategorija
- `MiB Mem` - ukupna i slobodna radna memorija u mebibajtovima (*MiB*), gdje je $1 \text{ MiB} = 2^{20}$ (1.048.576) bajtova.
- `MiB Swap` - ukupna i slobodna *swap* memorija u mebibajtovima (*MiB*)

Napomena: MebiByte nije isto što i Megabyte.

```
1 MB = 1,000,000 bajtova (B) = 1000 × 1000 B,
1 MiB = 1,048,576 B = 1024 × 1024 B.
```

Iako su obje jedinice vrlo slične i često se koriste naizmjenično u svakodnevnoj komunikaciji, postoji jasna tehnička razlika između njih. Megabyte (MB) koristi decimalni sustav (baza 10), dok Mebibyte (MiB) koristi binarni sustav (baza 2).

Mebibyte (MiB) i slične binarne jedinice (Kibibyte *KiB*, Gibibyte *GiB*, itd.) definirala je Međunarodna elektrotehnička komisija (IEC) još 1998. godine kako bi se izbjegla zabuna. Ove jedinice se češće koriste u Linux operativnim sustavima, gdje se preciznije prikazuje stvarna količina memorije. S druge strane, Windows, macOS i proizvođači hardvera uglavnom koriste decimalne jedinice – što nas dovodi do zanimljivog pitanja:

Jeste li se ikad pitali zašto dobijete manje memorije nego što ste platili?

Primjerice: Kupite 512 GB SSD, ali vam računalo pokazuje oko 475 "GB" slobodne memorije. "GB" je u ovom slučaju namjerno pod navodnim znakovima jer se ustvari radi o 475 GiB (Gibibyte). Razlog tome je što proizvođači hardvera iz marketinških (dakle finansijskih) razloga koriste decimalni sustav za prikazivanje kapaciteta, dok operacijski sustavi koriste binarni sustav.

Na primjer, `1 GiB = 1024 MiB`, dok `1 GB = 1000 MB`. Dakle, 512 GB je uistinu oko 476 GiB, i to nije

greška, već druga mjerna jedinica!

Ako pogledate prikaz procesa, uočite nekoliko važnijih stupaca:

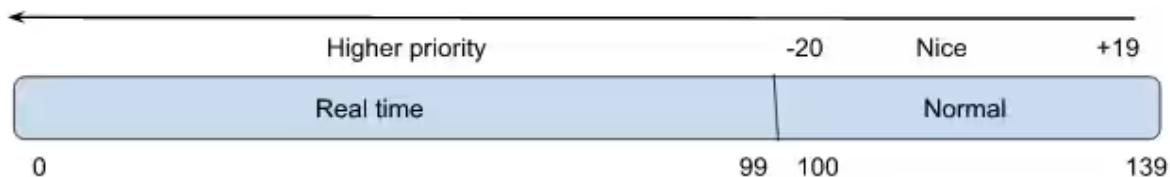
- **PID** - **jedinstveni identifikator procesa** (eng. *Process ID*)
- **USER** - **korisnik** koji pokreće proces
- **PR** - **prioritet procesa** (eng. *Process Priority*). **Veći broj znači manji prioritet.**
 - Na primjer, **20** je **manji prioritet** od **10**, a raspon je tipično od **0** do **20**.
 - Vrijednost **0** je **najviši prioritet**, dok je **20** najniži.
 - **PR** se za *user-space* (normalne) procese računa kao: $PR = 20 + NI$, gdje je **NI nice value** procesa.
- **NI** - "**nice value**" - još jedan "princip" određivanja prioriteta procesa.
 - Tipično se koristi za određivanje prioriteta procesa koji nisu *real-time* procesi, a iznosi između **-20** i **+19** gdje **-20** predstavlja najviši prioritet, a **+19** najniži.
- **VIRT** - ukupna **virtualna memorija** koju proces
- **RES** - **fizička memorija** koju proces koristi
- **SHR** - **dijeljena memorija** koju proces koristi
- **S** - **stanje procesa** (eng. *Process State*) - više o tome u nastavku
- **%CPU** - **postotak CPU-a** koji proces koristi
- **%MEM** - **postotak RAM-a** koji proces koristi
- **TIME+** - ukupno procesorsko vrijeme koje je proces koristio od pokretanja
- **COMMAND** - **naziv i argumenti naredbe koja je pokrenula proces**

Određivanje prioriteta procesa je važno jer OS koristi ovu vrijednost kako bi odredio koji procesi trebaju dobiti više resursa i procesorskog vremena.

 **Analogija:** Zamislimo računalo kao kuhinju u restoranu, a procese kao narudžbe koje treba pripremiti.

Recimo da su neke narudžbe super hitne (npr. za VIP goste), njih ćemo nazivati **real-time procesima**, dok su ostale narudžbe manje hitne i mogu čekati malo duže (npr. narudžbe za obične, non-VIP goste).

Sljedeća skala ilustrira **prioritetni raspon** (eng. *Priority range*) koji koristi Linux kernel za upravljanje procesima (skala se može razlikovati od distribucije do distribucije, ali generalno je slična):



 Prioritetni raspon koji koristi Linux kernel. Izvor: <https://blogs.oracle.com/linux/post/task-priority>

 **Napomena:** navedena skala predstavlja interni raspon prioriteta koji koristi Linux kernel, a ne korisnički prikaz koji vidimo u alatima poput **top** ili **htop**. Odnosno, ako naš proces ima **PR = 20**, to **ne znači** da je njegov prioritet **20** prema skali i da se radi o *real-time* procesu!

Uočite podjelu na skali:

- **Real-time** procesi (lijeva strana) imaju visoki prioritet - između 0 i 99, gdje 0 predstavlja najviši prioritet.
- **Normalni (nice)** procesi (desna strana) imaju niži prioritet - između 100 i 139, gdje 139 predstavlja najniži prioritet. Ovo su naši obični procesi (ili obični *nice* gosti kojima se ne žuri). Oni će čekati koliko god treba međutim, neki su malo nestrpljivi i žele da im se narudžba pripremi malo brže!
 - **nice** procesi imaju interni prioritet između -20 i +19, gdje -20 predstavlja goste (*taskove*) koji su nervozni i pomalo škrti po pitanju procesorske snage, dok +19 predstavlja drage goste koji su strpljivi i spremni su prepustiti procesorsku snagu drugim gostima.

Linux će uvijek prvo obraditi *real-time* procese.

Za *real-time* procese, NI se ne koristi.

Za normalne (tzv. *user-space*) procese, u pravilu se koristi formula za izračun prioriteta: PR = 20 + NI

Dakle:

- proces gdje je NI = 0 će imati PR = 20 (tzv. **zadani prioritet**)
- proces gdje je NI = 10 će imati PR = 30 (**manji prioritet od zadanog**)
- proces gdje NI = -20 će imati PR = 0 (**najveći prioritet među normalnim procesima**)

Uočit ćete da neki procesi imaju PR < 0, što najvjerojatnije predstavlja pogrešku u prikazu ili indikaciju da se radi o *real-time* procesu.

Alat htop

Moderniji alat koji se može koristiti za praćenje procesa je `htop`, koji je naprednija verzija `top` alata gdje su takve pogreške svedene na minimum. Osim toga, `htop` nudi i više opcija za filtriranje i sortiranje procesa, pristupačnije sučelje i obojeno sučelje, lakše pretraživanje i navigaciju kroz procese itd.

Sintaksa:

```
→ htop
```

The screenshot shows the htop command running on an Ubuntu Server. At the top, it displays system statistics: CPU usage (0.7% tasks, 22 threads, 1 running), memory usage (0.0% load average, 160M/1.91G total, 0K/1.87G free), and swap usage (0K/1.87G). Below this is a table of processes. The columns include PID, USER, PRI, NI, VIRT, RES, SHR, S, CPU%, MEM%, TIME+, and Command. The table lists various system daemons like /sbin/init, /usr/lib/systemd/systemd-journald, /sbin/multipathd, /usr/lib/systemd/systemd-udevd, and /usr/lib/systemd/systemd-timesyncd. The bottom of the window shows a menu bar with F1 through F10 keys and a status message about keyboard preferences.

Moderniji interaktivni prikaz aktivnih procesa koristeći naredbu `htop`.

Ako nije instaliran, možete ga instalirati ručno:

```
→ sudo apt install htop
```

Odmah možemo uočiti razliku kod prikaza prioriteta. Prioritet je ovdje prikazan kao `PRI`, a za *real-time* procese prikazuje se oznaka `RT` (*eng. Real Time*), a ne broj.

Rekli smo da je praktično koristiti `htop` jer pruža i jednostavnije sučelje za **filtriranje, sortiranje, pretraživanje** procesa itd.

- `F3` - pretraživanje procesa prema naredbi i argumentu (zadano ponašanje), primjer: `Search: python3`
- `F6` - sortiranje procesa prema različitim kriterijima, primjer: `SortBy: CPU%`
- `F4` - sortiranje procesa prema naredbi i argumentu (zadano ponašanje), primjer: `Filter: ModemManager`

The screenshot shows the htop command running on an Ubuntu Server. The terminal window title is "Ubuntu Server [Running]". The output displays system statistics at the top, including tasks, load average, and uptime. Below this is a table showing memory usage for processes. The table has columns: RI, NI, VIRT, RES, SHR, S, CPU%, MEM%, TIME+, and Command. A green bar at the top of the table indicates the current sort order by memory usage (MEM%). The command column shows two entries for "python3" processes.

RI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
20	0	105M	21632	12416	S	0.0	1.1	0:00.11	/usr/bin/python3 /usr/share/unattended-upgrades/unattended-
20	0	105M	21632	12416	S	0.0	1.1	0:00.00	/usr/bin/python3 /usr/share/unattended-upgrades/unattended-

At the bottom of the terminal window, there is a blue status bar with the text "Enter Done Esc Clear Filter: python3".

Primjer filtriranja `python3` procesa u `htop` alatu.

Što se tiče memorije, možemo uočiti 3 glavne kategorije:

1. `VIRT` - **virtualna memorija**
2. `RES` - **fizička memorija**
3. `SHR` - **dijeljena memorija**

1. Virtualna memorija (VIRT) predstavlja ukupnu količinu memoriju kojoj proces ima pristup, uključujući i memoriju koja nije fizički prisutna u sustavu. Ova vrijednost je često veća od fizičke memorije (RES) jer uključuje i **swap** memoriju, koja se koristi kada fizička memorija (RAM) postane nedovoljna.

- *swap* predstavlja dio diska koji se koristi kao proširenje fizičke memorije. Zamislite ju kao *backup* memoriju koju sustav koristi kada ponostaje fizičke memorije.
- *Primjer:* Kada koristimo web preglednik i otvorimo puno kartica, OS može neaktivne kartice "premjestiti" u swap memoriju kako bi oslobođio fizičku memoriju za aktivne kartice.
- Virtualna memorija uključuje i memoriju koju aktivni proces može koristiti, ali nije direktno prisutna u kompajliranom programu, npr. dijeljene biblioteke, konfiguracijske datoteke, druge OS-level datoteke itd.

2. Fizička memorija (RES) (eng. Resident Set Size / Physical Memory) predstavlja stvarnu količinu radne memorije (RAM) koju proces trenutno koristi. Drugim riječima, predstavlja količinu *non-swapped* memorije koju proces trenutno okupira u fizičkoj memoriji.

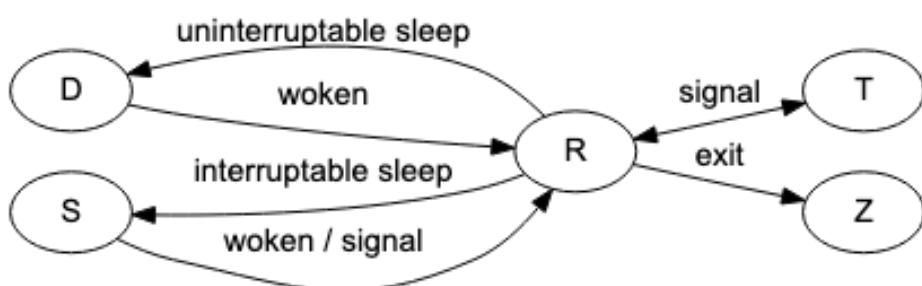
- ova vrijednost direktno utječe na performanse sustava i vrlo je važna metrika za praćenje.
- korespondira stupcu `%MEM` - postotak fizičke memorije koju proces koristi u odnosu na ukupnu fizičku memoriju sustava.

3. Dijeljena memorija (SHR) (eng. *Shared Memory*) predstavlja količinu fizičke (RES) memorije koju proces dijeli s drugim procesima u sustavu. Ova vrijednost je uvijek manja od fizičke memorije (RES) i može biti korisna za procjenu koliko memorije proces koristi u odnosu na druge procese.

Važan stupac koji nismo objasnili su stanja procesa - S.

Stanje procesa (eng. *Process State*) predstavlja trenutnu aktivnost procesa i može biti u **jednom od sljedećih stanja**:

- R - **Running** - proces se trenutno izvršava: aktivno koristi CPU
- S - **Sleeping** proces čeka na neki događaj (npr. I/O operaciju, signal ili semafor)
- D - **Uninterruptible Sleep** - proces čeka na I/O operaciju koja se ne može prekinuti (npr. čitanje s diska)
- T - **Stopped/Terminated** - proces je zaustavljen (npr. zbog signala - **CTRL+z**)
- Z - **Zombie** - proces je završen, ali ga i dalje možemo pronaći u procesnom prikazu (tablici), npr. zbog neodgovarajuće obrade signala od strane roditeljskog procesa
- X - **Dead** - proces je mrtav, potpuno je završen i ne može se više pronaći u procesnom prikazu. Ovu oznaku ćemo vrlo rijetko vidjeti.



Graf stanja procesa i prelazaka u drugo stanje. Izvor: <https://idea.popcount.org/2012-12-11-linux-process-states/>

Ako malo proscrollamo kroz procese, možemo vidjeti da gotovo svi procesi imaju oznaku S (sleeping). Ova oznaka označava stanje prekidljivog mirovanja, što ne znači da računalo ne obavlja nikakve aktivnosti. Naprotiv, riječ je o standardnom i učinkovitom mehanizmu upravljanja resursima u Linux operacijskom sustavu, kojim se omogućuje da procesi čekaju događaje bez nepotrebnog opterećenja procesora.

Uočit ćete i jedan aktivan proces (R) - a to je upravo naš htop alat koji aktivno koristi CPU.

Vježba: Sada kada smo usvojili kako pronaći i tumačiti procese, vrijeme je za malo praktične primjene. U nastavku ćemo analizirati nekoliko konkretnih procesa kako bismo bolje razumjeli njihovo ponašanje i ulogu u sustavu:

Proces htop

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	COMMAND
1041	lukablaskovic	20	0	5512	3712	2944	R	1.3	0.2	0:15.01	htop

- PID 1041 - jedinstveni identifikator procesa, pokreće ga korisnik lukablaskovic
- Prioritet je 20, a nice value je 0. Dakle, radi se o normalnom procesu (ne real-time) koji ima **zadani prioritet**.

- Proces koristi ukupno 5512 B virtualne memorije, od čega 3712 B fizičke memorije. Veliki dio fizičke memorije je dijeljen s drugim procesima (2944 B).
 - Proces je trenutno aktivan (R), zauzima 1.3% CPU-a i 0.2% fizičke memorije. Procesor je koristio ukupno ~15 minuta od pokretanja.
 - Naredba koja je pokrenula proces je htop .
-

Proces -bash

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	COMMAND
986	lukablaskovic	20	0	5780	4736	3200	S	0.0	0.2	0:00:04	-bash

- PID 986 - jedinstveni identifikator procesa, pokreće ga korisnik lukablaskovic
 - Prioritet je 20 , a nice value je 0 . Dakle, radi se o normalnom procesu (ne real-time) koji ima zadani prioritet.
 - Proces koristi ukupno 5780 B virtualne memorije, od čega 4736 B fizičke memorije. Veliki dio fizičke memorije je dijeljen s drugim procesima (3200 B).
 - Proces trenutno spava (S), ne zauzima CPU, ali koristi 0.2% fizičke memorije. Procesor je koristio ukupno ~4 sekunde od pokretanja.
 - Naredba koja je pokrenula proces je -bash .
-

Proces multipathd

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	COMMAND
340	root	RT	0	347M	26240	7296	S	0.0	1.3	0:00:00	/sbin/multipathd -d -s

- PID 340 - jedinstveni identifikator procesa, pokreće ga korisnik root odnosno superkorisnik
 - Radi se o real-time procesu što je jasno iz oznake RT , a NI je 0 .
 - Proces koristi ukupno 347 MB virtualne memorije, od čega 26240 B fizičke memorije (~0.02 MB), što indicira da se radi o procesu koji koristi jako puno virtualne memorije, ali vrlo malo fizičke memorije. Manji dio fizičke memorije je dijeljen s drugim procesima (7296 B).
 - Proces trenutno spava (S), ne zauzima CPU, ali koristi 1.3% fizičke memorije. Procesor je koristio ukupno ~0 sekundi od pokretanja.
 - Naredba koja je pokrenula proces je /sbin/multipathd -d -s .
-

Zadatak 1: Tumačenje procesa

Otvorite `htop` alat i analizirajte barem:

- 1 *real-time* proces
- 1 *normalni* proces koji je pokrenuo korisnik `root`
- 1 *normalni* proces koji ste vi pokrenuli

Protumačite sve stupce kao što smo to napravili u prethodnim primjerima.

1.2 Programske dretve

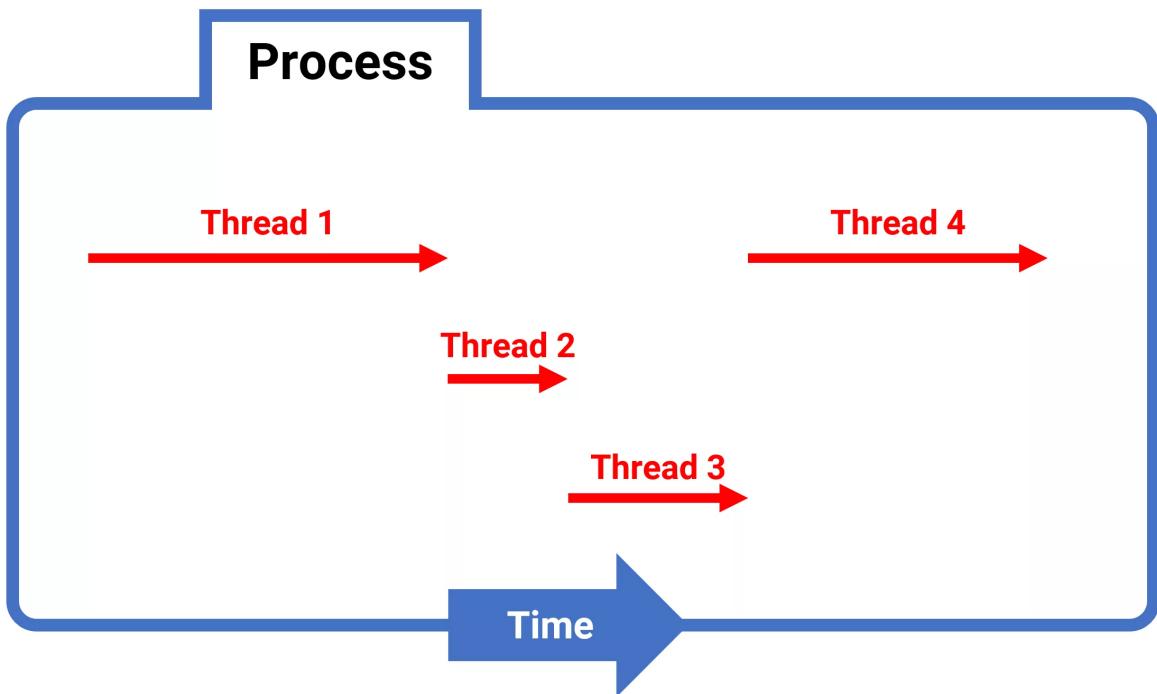
Dretva (*eng. thread*) je **najmanja jedinica izvršavanja** unutar procesa. Više dretvi može postojati unutar istog procesa, a svaka dretva može samostalno izvršavati skup instrukcija. Sve dretve jednog procesa **dijele isti adresni prostor** (memoriju), uključujući varijable, otvorene datoteke i druge resurse. Međutim, svaka dretva ima **vlastiti programski brojač, registarski skup i vlastiti stog** (*eng. stack*), što joj omogućuje neovisno izvršavanje.

To omogućuje **višezadaćnost unutar jednog procesa** (*tzv. multithreading*), gdje više dretvi može istodobno raditi na različitim zadacima, često s boljom učinkovitošću nego pokretanje više procesa, jer je međusobna komunikacija između dretvi brža i manje zahtjevna.

S druge strane, proces je samostalna instanca programa u izvršavanju, s vlastitim adresnim prostorom i resursima. Dretve unutar istog procesa surađuju na izvršavanju zadataka dijeleći kontekst procesa.

Primjer: Moderni web preglednici koriste **višestruke procese** (*multiprocessing*) i višestruke dretve (*multithreading*) za postizanje bolje stabilnosti i performansi.

- Svaka kartica u pregledniku često se otvara kao zaseban proces, što povećava stabilnost i sigurnost — ako dođe do pada jedne kartice, ostale nisu pogodjene.
- Unutar svake kartice (procesa), preglednik koristi više dretvi za paralelno izvršavanje različitih zadataka kao što su:
 - učitavanje i prikaz HTML sadržaja
 - obrada i izvršavanje JavaScript koda
 - mrežne operacije (npr. dohvaćanje resursa)
 - prikazivanje slika i videozapisa
 - upravljanje korisničkim sučeljem

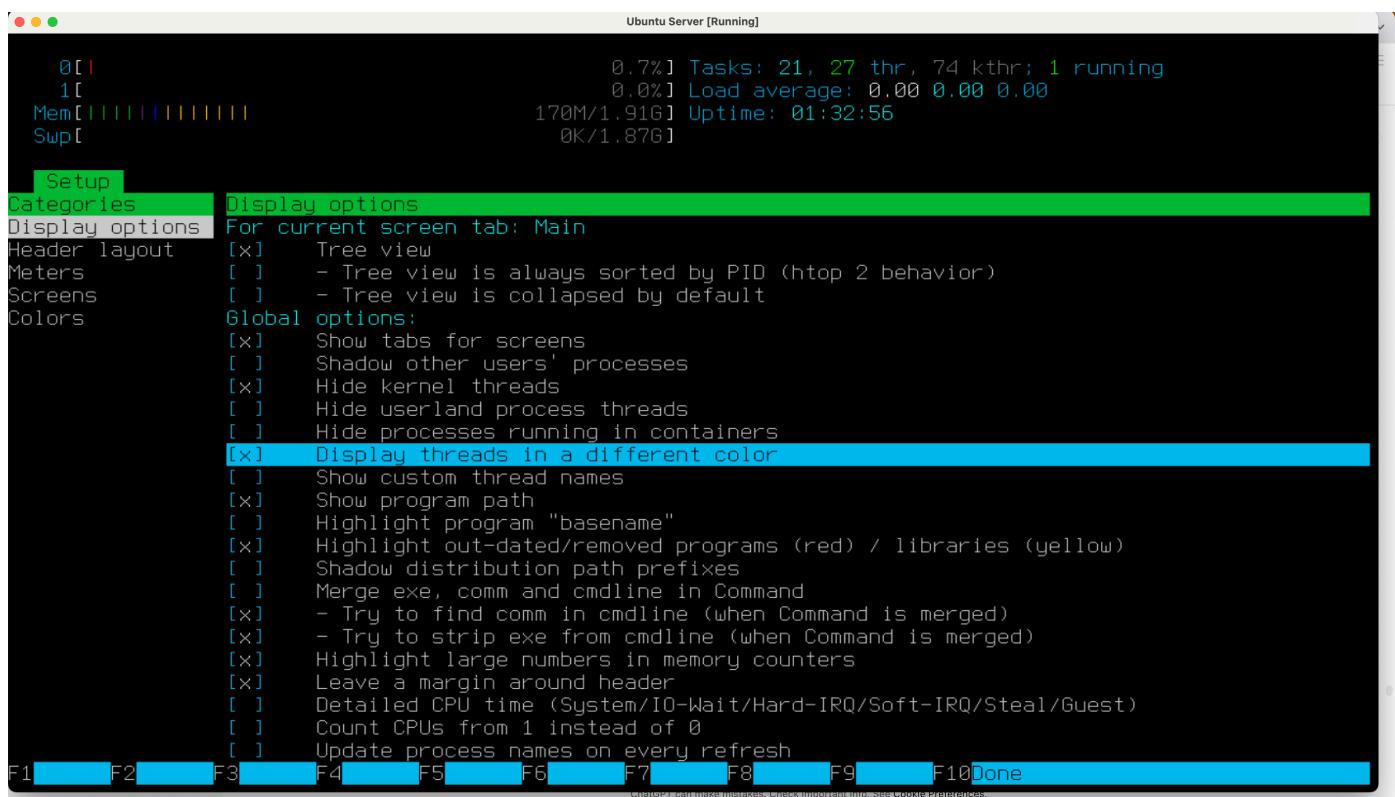


Ilustracija prikazuje više dretvi unutar istog procesa, od kojih svaka ima različito trajanje izvršavanja tijekom rada procesa.

Dretveni prikaz koristeći `htop` će najvjerojatnije biti zadan, ako nije, možemo ga pokrenuti zastavicom `-H`:

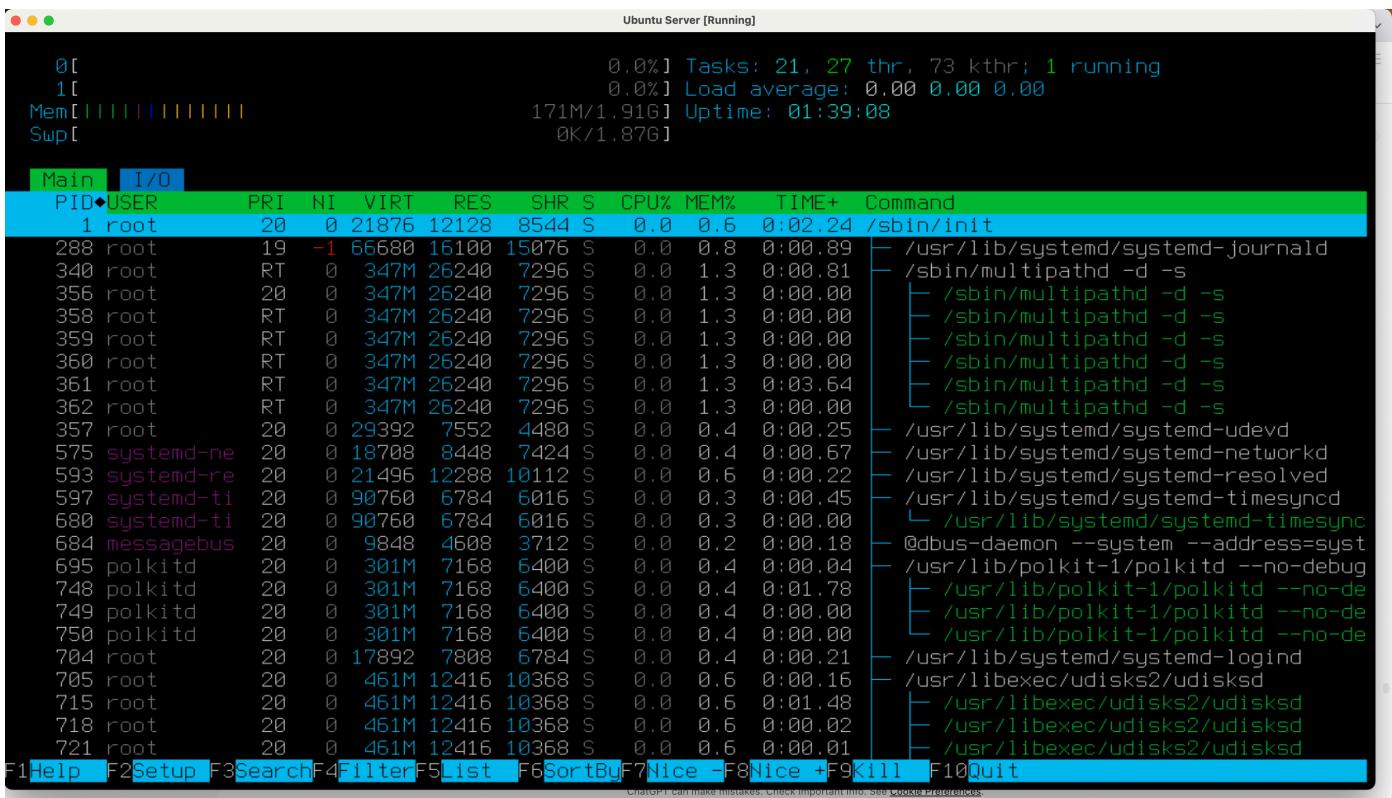
```
→ htop -H
```

Možemo otvoriti `Setup (F2)` i omogućiti opciju `Display options -> Display threads in a different color` kako bi lakše razlikovali dretve od procesa.



 htop: Setup F2 -> Display options -> Display threads in a different color

Pod `command` čete sada uočiti dretve definirane **zelenom bojom**. Dodatno, možemo prikazati hijerarhiju pritiskom na tipku `F5` (*Tree view*).



 htop: Prikaz hijerarhije procesa i dretvi, hijerarhijski odabirom `F5` opcije

Primjerice, za proces `/sbin/multipathd -d -s` možemo vidjeti da se radi o procesu koji koristi 6 dretvi. Dodatno, u ovom slučaju dretve dijele resurse pa možemo vidjeti iste vrijednosti za `VIRT`, `RES` i `SHR` i `MEM%` stupce.

Općenito, unutar `htop` alata možemo u gornjem desnom kutu vidjeti ukupan **broj procesa, dretvi i skrivenih kernel dretvi**.

- `Tasks` - **ukupan broj procesa**. Na slici iznad je 21, a taj broj možete dobiti prebrojavanjem procesa u `htop` alatu koji su označeni bijelom bojom.

Uočite sljedeće: Kod procesa `/bin/login -p` nemamo dretve, već potprocese `-bash` i `htop -H`. Ovakva hijerarhija se može dogoditi kada proces pokrene drugi proces, `bash` i `htop` su potprocesi koji su pokrenuti iz procesa `/bin/login -p`, a ne dretve.

- `threads (thr)` - označeno zelenom bojom, predstavlja **ukupan broj dretvi** koje različiti procesi koriste. Na slici iznad je 27, a taj broj možete dobiti prebrojavanjem dretvi u `htop` alatu koji su označene zelenom bojom.
- `hidden kernel threads` - označeno sivom bojom, predstavlja **ukupan broj skrivenih kernel dretvi**. Na slici iznad je 74. Ne možemo ih prebrojati jer su skrivene, ali ih možemo prikazati tako što otvorimo `Setup (F2)` i onemogućimo opciju `Display options -> Hide kernel threads`.

Kada omogućimo skrivene kernel dretve, možemo ih vidjeti u `htop` alatu označene isto zelenom bojom.



: Prikaz skrivenih kernel dretvi - označene isto tako zelenom bojom

Alat `kill`

Kako bismo prekinuli neki proces (ili dretvu), možemo koristiti alat `kill`. Ova naredba šalje određeni signal procesu kako bi ga prekinula ili promijenila njegovo stanje.

Naredba `kill` koristi se za slanje signala procesu putem njegovog PID-a, i ne mora se raditi o prekidu procesa, već i o drugim signalima koji mogu promijeniti njegovo ponašanje. Međutim, najčešće se koristi za prekid procesa interruptivnim signalom koji prebacuje stanje procesa iz `R` (Running) u `T` (stopped).

Sintaksa:

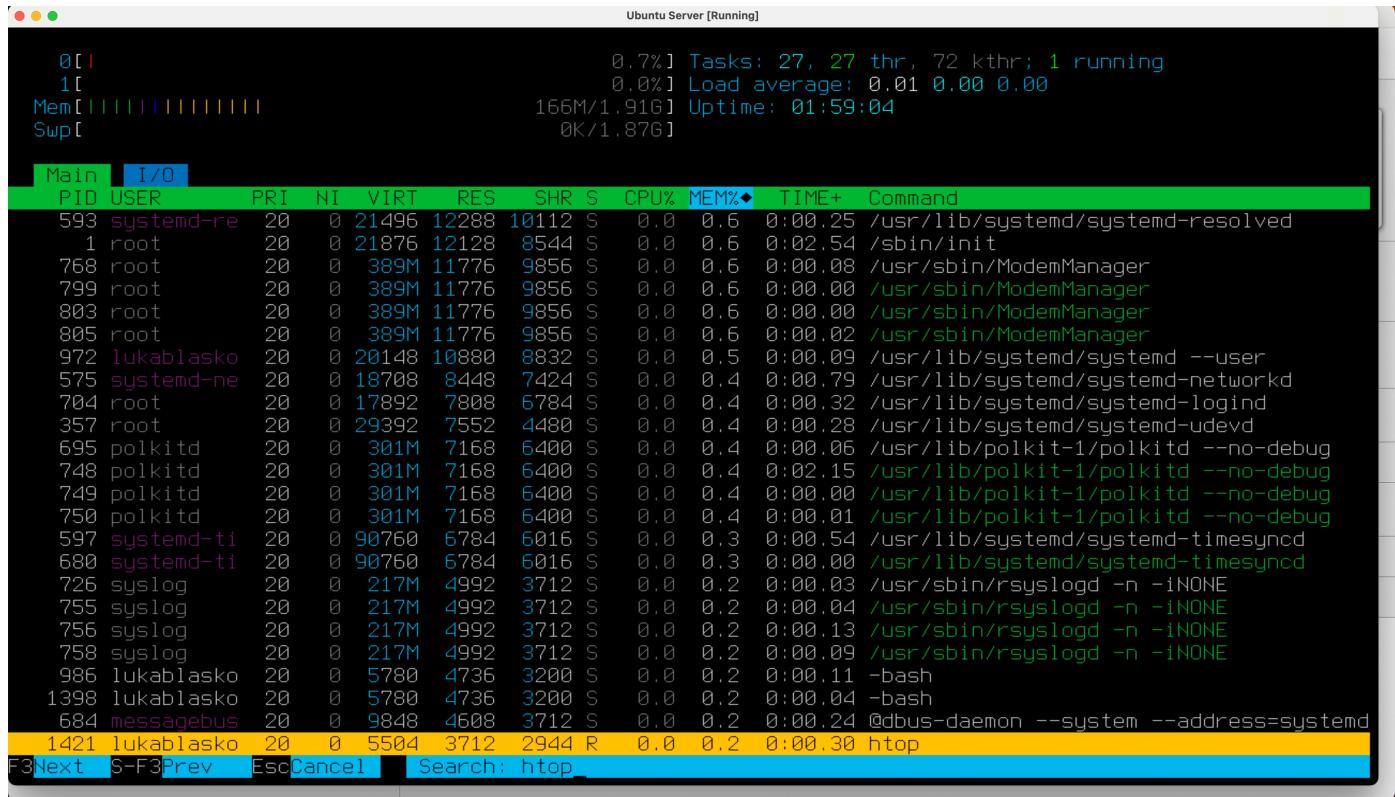
```
→ kill [opcije/zastavice] <PID>
```

Opcije (zastavice) predstavljaju vrstu signala koji šaljemo procesu. Najčešće korišteni signali su:

- `-9 - SIGKILL` - prisilno prekida proces, ne može se obraditi
- `-15 - SIGTERM` - šalje procesu zahtjev za prekidom, proces može obraditi signal i završiti se na "ljudazan" način
- `-1 - SIGHUP` - šalje procesu signal za ponovno učitavanje konfiguracije, često se koristi za ponovno učitavanje usluga
- `-2 - SIGINT` - šalje procesu signal za prekid, obično se koristi kada pritisnemo `CTRL+C` u terminalu
- `-19 - SIGSTOP` - šalje procesu signal za pauziranje, proces se ne može obraditi i ne može se nastaviti dok ga ne prekinemo
- `-18 - SIGCONT` - šalje procesu signal za nastavak nakon pauziranja

Prema zadanim postavkama, `kill` šalje signal `SIGTERM` (`-15`) procesu.

Otvorite `htop` i pronađite `PID` procesa `htop`. Možete i pretraživati (`F3`) kako biste ga lakše pronašli.



`htop`: Pronalazak procesa pretraživanjem po nazivu (`F3`)

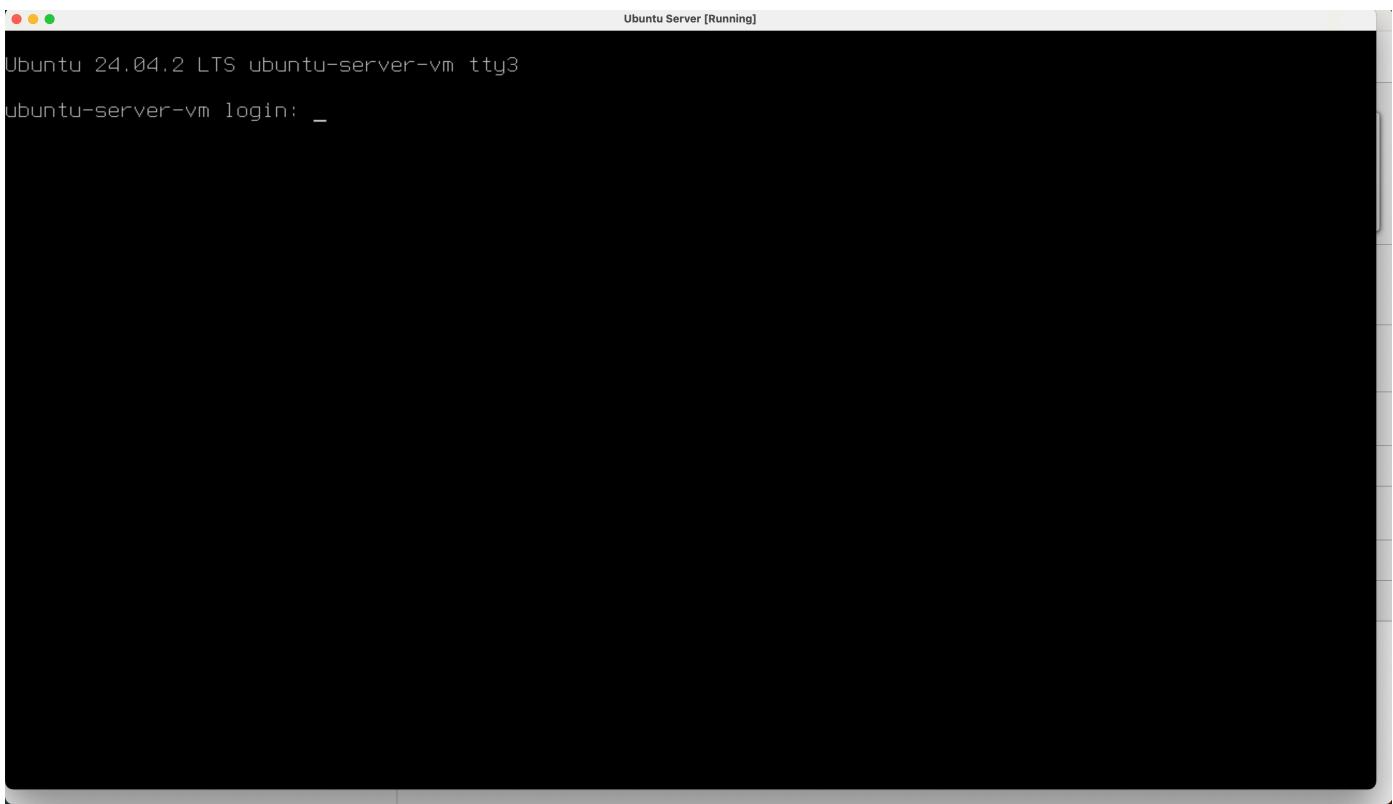
Vidimo da je `PID` našeg `htop` procesa `1421`. Dakle, prekinuli bismo ga koristeći:

```
→ kill 1421  
# ili agresivno  
→ kill -9 1421
```

Problem: `htop` proces zauzima terminal sučelje i ne možemo upisivati druge bash naredbe.

Možemo se spojiti sa SSH klijentom na naš virtualni stroj i pozvati naredbu u novoj sesiji, ili pak možemo pokretati novu sesiju pritiskom `ALT + →` ili `ALT + ←` i na taj način otvoriti novu **Virtualnu konzolu** (`TTY`):

Trebali biste vidjeti nove sesije koje razlikujete po oznaci `tty[N]`, npr. `tty1`, `tty2`, `tty3` itd. `tty` je stara oznaka za terminal sučelje, a `N` predstavlja broj terminala.



 Nova terminal sesija (`tty3`) u koju smo se prebacili pritiskom na `ALT + →`

Prijavite se i pokrenite naredbu za prekidanje procesa `htop`:

```
→ kill 1421
```

Prebacite se natrag u prethodnu sesiju, vidjet ćete da je `htop` proces prestao raditi - terminiran je.

Alati `pidof` i `pgrep`

`PID` procesa, osim pretraživanjem unutar `htop` alata, možemo pronaći i pomoću naredbi `pidof` ili `pgrep`:

- Ako proces nije pokrenut ili je unesen netočan naziv, naredba `pidof` neće prikazati nikakav izlaz/output
- možemo se i uvjeriti ispisom statusa posljednje naredbe (`$?`)

```
→ pidof htop # ispisuje PID procesa "htop" ako je pokrenut
```

```
→ echo $? # ispisuje 0 ako je proces pokrenut i postoji PID, inače ispisuje 1
```

Vrlo često nećemo znati točan naziv procesa pa je praktično koristiti i pretraživanje po najbližem podudaranju - koristeći alat `pgrep` baziran na `grep`-u:

Recimo, `htop` možemo pretraživati kao `ht` ili `hto`:

```
→ pgrep htop # ispisuje PID procesa "htop" ako je pokrenut  
→ echo $? # ispisuje 0 ako je proces pokrenut i postoji PID, inače ispisuje 1  
  
→ pgrep Modem # ispisuje PID procesa "ModemManager" ako je pokrenut  
→ pidof Modem # Neće ispisati ništa jer je puni naziv naredbe koja pokreće proces  
"ModemManager"
```

Alat `pkill`

Postoji skraćena verzija `kill` naredbe koja se koristi za zaustavljanje procesa prema nazivu, a ne prema `PID`-u. Ova naredba se zove `pkill` i na neki način je kombinacija naredbi `pgrep` i `kill`.

Sintaksa:

```
→ pkill [opcije/zastavice] <ime_procesa>
```

Na primjer, ako želimo prekinuti sve procese koji se zovu `htop`, možemo koristiti:

```
→ pkill htop
```

Postoji i varijanta - `killall` koja se koristi za prekid svih procesa koji imaju isto ime, bez obzira na korisnika koji ih pokreće. Primjerice, želimo odjednom zatvoriti `chrome` ili `firefox` i sve njihove procese:

```
→ killall chrome  
→ killall firefox
```

Ova naredba će neizravno prekinuti i sve dretve koje su pokrenute unutar tih procesa, budući da su dio tog procesa.

1.3 Alat `nice`

Alat `nice` se koristi za pokretanje procesa s definiranim `NI` prioritetom. Ova naredba omogućava korisnicima da odrede prioritet procesa koji pokreću, čime se može utjecati na raspodjelu resursa i CPU vremena između različitih procesa.

Rekli smo da je `NI` prioritet procesa između `-20` i `19`, gdje `-20` predstavlja najviši prioritet, a `19` najniži.

Sintaksa:

```
→ nice -n <prioritet> <naredba>
```

- `<prioritet>` - predstavlja prioritet koji želimo postaviti procesu, a može biti između `-20` i `19`
- `<naredba>` - predstavlja naredbu koju želimo pokrenuti s određenim prioritetom (može biti bilo koja CLI naredba, uključujući i bash izraz ili skripta)

Primjer: Ako definiramo bash skriptu `numbers.sh` koja ispisuje brojeve od `1` do `100`, možemo ju pokrenuti s različitim prioritetima:

```
→ nano numbers.sh
#!/bin/bash

for i in {1..100}
do
    echo $i
done
```

Dodjeljujemo dozvolu za izvršavanje `x`:

```
→ chmod +x numbers.sh
```

Skriptu možemo pokrenuti s različitim prioritetima:

```
→ nice -n 0 ./numbers.sh # pokreće skriptu s prioritetom 0 (zadani prioritet)

→ nice -n 10 ./numbers.sh # pokreće skriptu s prioritetom 10 (manji prioritet od zadanog)

→ sudo nice -n -10 ./numbers.sh # pokreće skriptu s prioritetom -10 (veći prioritet od zadanog)
```

Ponekad je za pokretanje naredbi s većim prioritetom potreban `sudo` pristup. Ako pokušamo pokrenuti skriptu s višim prioritetom bez `sudo` naredbe, dobit ćemo poruku:

```
nice: cannot set niceness: Permission denied
```

Skripta radi, međutim ako se paralelno u drugom terminalu prebacimo u `htop`, nećemo vidjeti naš proces jer se prebrzo izvrši i proces tu nestaje.

Naredbom `sleep` ćemo usporiti ispisivanje brojeva, nakon svakog ispisa ćemo čekati `1` sekundu:

```
#!/bin/bash

for i in {1..100}
do
    echo $i
    sleep 1
done
```

Ako jednostavno pokrenemo skriptu, ona će se ustvari pokrenuti sa zadanim prioritetom, odnosno:

- `NI = 0`
- `PRI = 20`

```
→ ./numbers.sh
```

Pokrenimo `htop` i pogledajmo kako se proces ponaša:

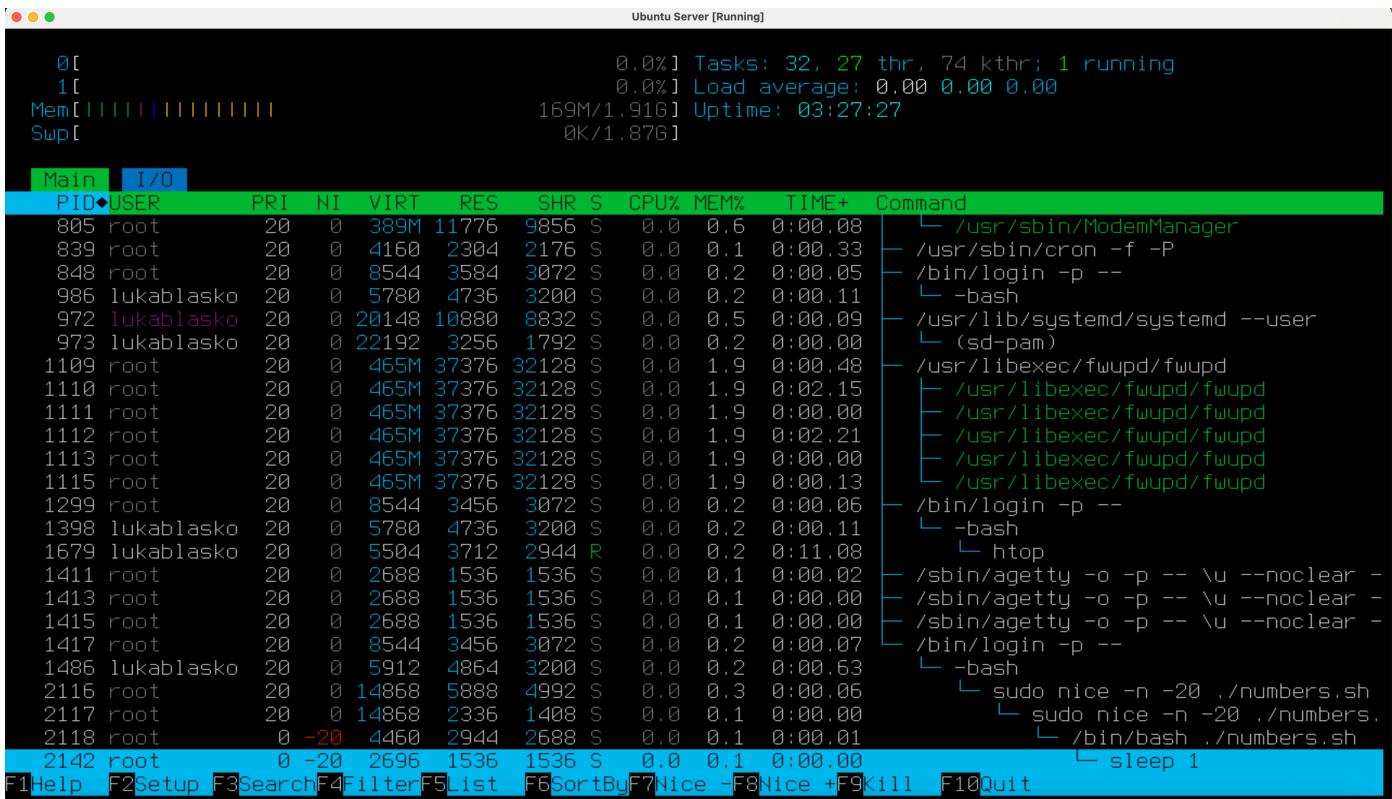
Uočite sljedeću hijerarhiju procesa/potprocesa:

- `/bin/login -p --` - proces koji se pokreće na početku (login sesija)
 - `-bash` - potproces koji pokreće `bash` shell
 - `/bin/bash .numbers.sh` - potproces koji pokreće našu bash skriptu
 - `sleep 1` - potproces koji pokreće `sleep` naredbu

Ako pokrenemo skriptu s `nice` naredbom, primijetite da se procesi nalaze u drugoj hijerarhiji:

```
→ nice -n 10 ./numbers.sh # pokreće skriptu s prioritetom 10 (manji prioritet od zadanog)

→ sudo nice -n -20 ./numbers.sh # pokreće skriptu s prioritetom -20 (najviši prioritet)
```



`htop` prikaz: Proces instanciran naredbom `nice` s najvišim mogućim prioritetom od `-20`.

Ako želimo promijeniti prioritet već pokrenut procesa (u fazi `R`), možemo koristiti `renice` varijantu naredbe:

Sintaksa:

```
→ renice -n <prioritet> -p <PID>
```

Na primjer, ako želimo povećati prioritet izvođenja procesa `./numbers.sh`, možemo prvo pronaći njegov `PID` koristeći `htop` ili `pidof` ili `pgrep`:

```
→ pgrep numbers # vraća PID pokrenutog procesa
```

 **Napomena** : puni naziv procesa je: `/bin/bash ./numbers.sh` zbog shebang oznake na početku skripte

Smanjiti ćemo mu prioritet na `-10`:

```
→ sudo renice -n -10 -p <PID>

# Primjer:
→ sudo renice -n -10 -p 2351
```

Otvorite alat `htop` i provjerite je li se prioritet izvođenja procesa promijenio.

Zadatak 2: Upravljanje procesima

Alat `node` omogućuje pokretanje JavaScript koda izvan okruženja web preglednika. Možete ga jednostavno instalirati u Ubuntu Server koristeći `apt` alat. Jednom kada ga instalirate, u `home` direktoriju stvorite novu datoteku `stopwatch.js` koja će pauzirati izvršavanje procesa koristeći `prompt` funkciju. Kroz `prompt` funkciju korisnik unosi broj sekundi koje moraju isteći prije nego što se proces završi.

U JavaScriptu, `timer` možete implementirati koristeći `setTimeout` funkciju koja prima `callback` funkciju (ono što se poziva jednom kad vrijeme istekne) i vrijeme u milisekundama (koliko se čeka):

```
setTimeout(callback_fn, vrijeme_ms);
```

Jednom kad napišete skriptu, možete ju pokrenuti koristeći naredbu `node`.

Pokrenite skriptu i provjerite kako se proces ponaša koristeći `htop` alat. Ispišite PID procesa i provjerite njegovu hijerarhiju.

Pokrenite skriptu koristeći `nice` naredbu sa:

- zadanim prioritetom
- prioritetom `-10`
- prioritetom `10`

`node` omogućuje pristup procesnim informacijama putem `process` objekta u JS kodu. Na primjer, možete ispisati PID trenutnog procesa koristeći:

```
console.log(process.pid);
```

Ispišite PID trenutnog procesa jednom kad on započne, zatim pomoću `htop` alata provjerite podudara li se.

2. Upravljanje korisnicima

Upravljanje korisnicima i grupama (*eng. User and Group Management*) još je jedan ključan aspekt administracije Linux sustava, koji je, srećom, znatno jednostavniji od upravljanja procesima.

Cijela priča svodi se na svega nekoliko osnovnih naredbi koje možemo koristiti za dodavanja, uklanjanje i upravljanje korisnicima/grupama.

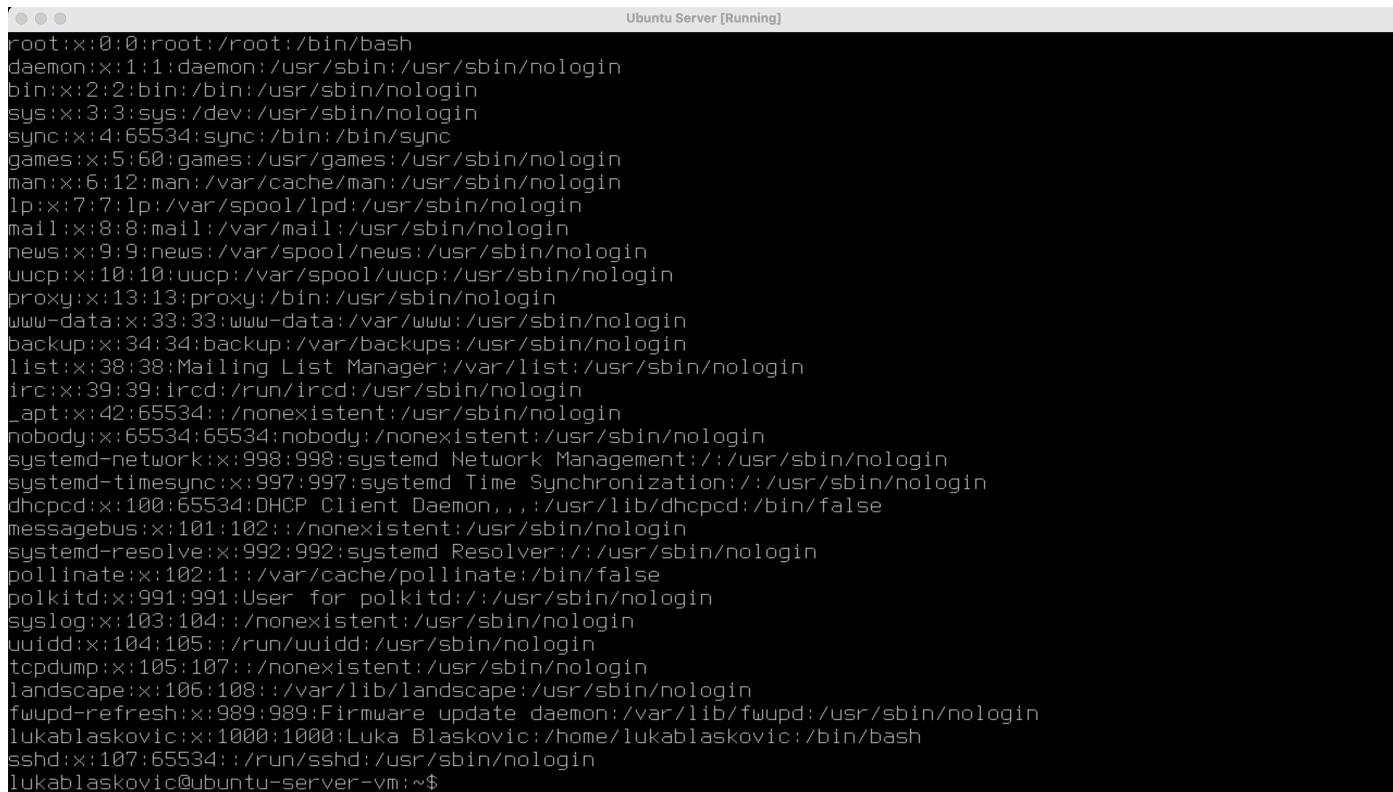
Prije nego što krenemo manipulacijama korisnika, nije loše provjeriti aktivne sesije i aktivnog (prijavljenog) korisnika, naredbama `who` odnosno `whoami`:

```
→ who # ispisuje sve aktivne korisnike i njihove terminal sesije (TTY1, TTY2, TTY3, itd.)  
→ whoami # ispisuje trenutnog korisnika koji je prijavljen
```

Kako bismo zatvorili one prethodne sesije, možemo izaći pomoću `exit` ili `CTRL + D`.

Kako bismo provjerili **sve pohranjene korisnike**, možemo pročitati sadržaj datoteke `/etc/passwd`:

```
→ cat /etc/passwd
```



```
Ubuntu Server [Running]  
root:x:0:0:root:/root:/bin/bash  
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin  
bin:x:2:2:bin:/bin:/usr/sbin/nologin  
sys:x:3:3:sys:/dev:/usr/sbin/nologin  
sync:x:4:65534:sync:/bin:/bin/sync  
games:x:5:60:games:/usr/games:/usr/sbin/nologin  
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin  
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin  
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin  
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin  
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin  
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin  
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin  
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin  
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin  
irc:x:39:ircd:/run/ircd:/usr/sbin/nologin  
_apt:x:42:65534::/nonexistent:/usr/sbin/nologin  
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin  
systemd-network:x:998:998:systemd Network Management:/:/usr/sbin/nologin  
systemd-timesync:x:997:997:systemd Time Synchronization:/:/usr/sbin/nologin  
dhcpcd:x:100:65534:DHCP Client Daemon,,,,:/usr/lib/dhcpcd:/bin/false  
messagebus:x:101:102::/nonexistent:/usr/sbin/nologin  
systemd-resolve:x:992:992:systemd Resolver:/:/usr/sbin/nologin  
pollinate:x:102:1::/var/cache/pollinate:/bin/false  
polkitd:x:991:991:User for polkitd:/:/usr/sbin/nologin  
syslog:x:103:104::/nonexistent:/usr/sbin/nologin  
uuidd:x:104:105::/run/uuidd:/usr/sbin/nologin  
tcpdump:x:105:107::/nonexistent:/usr/sbin/nologin  
landscape:x:106:108::/var/lib/landscape:/usr/sbin/nologin  
fwupd-refresh:x:989:989:Firmware update daemon:/var/lib/fwupd:/usr/sbin/nologin  
lukablaskovic:x:1000:1000:Luka Blaskovic:/home/lukablaskovic:/bin/bash  
sshd:x:107:65534::/run/sshd:/usr/sbin/nologin  
lukablaskovic@ubuntu-server-vm:~$
```

 Prikaz svih korisnika u konfiguracijskoj datoteci `/etc/passwd`

Ova datoteka sadrži sve korisnike koji su registrirani, svaki redak predstavlja jednog korisnika.

Svaki zapis je u formatu:

```
username:x:UID:GID:comment:home_directory:shell
```

- `username` - korisničko ime
- `x` - oznaka za lozinku (nekad je ovdje bila pohranjena lozinka, sada je pohranjena u konf. datoteci: `/etc/shadow`)
- `UID` - jedinstveni identifikator korisnika (User ID)
- `GID` - jedinstveni identifikator grupe (Group ID)
- `comment` - opis korisnika (npr. može biti puno ime i prezime)
- `home_directory` - putanja do korisničkog direktorija (npr. `/home/lukablaskovic`)
- `shell` - zadani shell koji se koristi prilikom prijave (npr. `/bin/bash`)

Vidjet ćete veliki broj korisnika, uz vašeg koji se nalazi pri dnu. Radi se o sustavnim korisnicima (eng. system users) koji su potrebni za rad sustava i raznih servisa.

Na vrhu ćete vidjeti `root` korisnika koji je **superkorisnik i ima sve privilegije u sustavu**.

Kako bismo se prebacili u `root` korisnika, možemo koristiti naredbu:

```
→ sudo -i
```

```
lukablaskovic@ubuntu-server-vm:~$ sudo -i
[sudo] password for lukablaskovic:
root@ubuntu-server-vm:~# echo "Sada sam superkorisnik i ne moram stalno pisati sudo"
Sada sam superkorisnik i ne moram stalno pisati sudo
root@ubuntu-server-vm:~#
```

Prebacivanje u `root` korisnika koristeći `sudo -i` naredbu

Kada smo u `root` korisniku, možemo koristiti sve privilegije i upravljati sustavom bez ograničenja - **ne moramo koristiti `sudo` ispred svake naredbe**.

Kako bismo se prebacili natrag u našeg korisnika, možemo koristiti naredbu `su`:

Sintaksa:

```
→ sudo su - <username>
```

- gdje je `<username>` korisničko ime korisnika u kojem se prebacujemo

Na ovaj način se možemo prebaciti i u `root` korisnika:

```
→ sudo su - root # isto što i sudo -i
```

2.1 Naredba `useradd`

Naredba `useradd` služi za kreiranje novih korisničkih računa u sustavu. Za njezino izvođenje potrebne su administratorske ovlasti (privilegije *superusera* ili pristup putem korisnika `root`).

Prilikom dodavanja korisnika, sustav automatski ažurira ključne **konfiguracijske datoteke**:

- `/etc/passwd` (sadrži osnovne informacije o korisnicima: korisničko ime, UID, GID, home direktorij, zadani shell, itd.)
- `/etc/shadow` (sadrži *hashirane* lozinke i informacije o isteku lozinki, ako postoje)
- `/etc/group` (sadrži informacije o grupama i članovima grupa)
- `/etc/gshadow` (sadrži *hashirane* lozinke za grupe i informacije o članovima grupa)

 **Napomena:** Upravljanje korisnicima je teoretski moguće izvoditi i direktnom izmjenom ovih konfiguracijskih datoteka, međutim to nije preporučljivo jer je lako napraviti pogrešku

Sintaksa:

```
→ useradd [opcije/zastavice] <username>
```

- `<username>` - korisničko ime korisnika kojeg dodajemo

Primjer: Dodajemo novog korisnika `markomaric`:

```
→ sudo useradd markomaric
```

Prema zadanim postavkama, novi korisnik će biti bez lozinke, a njegov home direktorij će biti `/home/markomaric`. Također, zadani shell će biti `/bin/bash`. Isto možemo provjeriti u datoteci `/etc/passwd`:

```
sudo cat /etc/passwd
```

 **Hint:** Općenito, kada čitamo velike datoteke nije loše preusmjeriti njihov *output* u naredbu `grep`. Naredba `grep` (global regular expression print) je utility alat koji se koristi za pretraživanje *plaintext* sadržaja pomoću nekog regularnog izraza (*eng. regular expression*).

Preusmjeravanje (*eng. piping*) izvodimo pomoću znaka `|` (pipe). Zapamti kao okomitu cijev.

Sintaksa:

```
→ naredba | naredba_2 | naredba_3
```

Na primjer, ako želimo brzo pronaći korisnika `markomaric` u datoteci `/etc/passwd`, možemo koristiti:

```
→ cat /etc/passwd | grep markomaric # preusmjeri rezultat naredbe cat u naredbu grep (koja ga obrađuje)
```



```
Ubuntu Server [Running]
lukablaskovic@ubuntu-server-vm:~$ cat /etc/passwd | grep markomaric
markomaric:x:1001:1001::/home/markomaric:/bin/sh
lukablaskovic@ubuntu-server-vm:~$
```

 Preusmjeravanje ukupnog sadržaja iz datoteke `/etc/passwd` u `grep` alat koji filtrira sadržaj prema regularnom izrazu, konkretno izraz je običan string: `markomaric`

Ipak, naredba `useradd` neće automatski stvoriti *home* direktorij za novog korisnika, već to moramo učiniti sami. Prebacit ćemo se u korisnika `markomaric` i pokušati mu stvoriti *home* direktorij:

```
→ sudo su - markomaric # prebacujemo se u novog korisnika
→ whoami # markomaric
→ mkdir /home/markomaric # stvaramo home direktorij
```

Međutim dobit ćemo upozorenje da korisnik nema *home* direktorij:

```
su: warning: cannot change directory to /home/markomaric: No such file or directory
```

Upišite `exit` kako biste se vratili u vašeg korisnika.

Stvorit ćemo direktorij ručno:

```
→ whoami # lukablaskovic
→ sudo mkdir /home/markomaric
→ sudo su - markomaric # prebacujemo se u novog korisnika
```

Ipak, ako pokušate išta raditi s novim korisnikom, nećete moći jer `markomaric` nema privilegije.

Ovo nije preporučen način! Ne želimo nikad ručno stvarati *home* direktorij, već koristiti odgovarajuću zastavicu/zastavice prilikom pozivanja `useradd` naredbe.

Pogledat ćemo nekoliko korisnih zastavica kako bismo inicijalno korisniku podesili neke postavke, poput *home* direktorija, grupa, zadanog shella i dr.

<code>useradd</code> zastavica	Opis zastavice	Primjer
<code>-m</code>	Stvara korisnički <i>home</i> direktorij (<code>/home/ime</code>)	<code>useradd -m anaanic</code>
<code>-d DIR</code>	Postavlja <i>custom home</i> direktorij na <code>DIR</code> , ako nije navedeno postavlja se na zadani <code>/home/korisnik</code>	<code>useradd -m -d /opt/anaanic anaanic</code>
<code>-s SHELL</code>	Postavlja korisniku zadani shell (<code>SHELL</code>)	<code>useradd -m -s /bin/bash anaanic</code>
<code>-u UID</code>	Postavlja korisnički ID (<code>UID</code>)	<code>useradd -m -u 1050 anaanic</code>
<code>-g GROUP_NAME</code>	Postavlja primarnu grupu korisnika na <code>GROUP_NAME</code> . U nastavku više o grupama.	<code>useradd -m -g developeri anaanic</code>
<code>-G GROUP1, GROUP2</code>	Postavlja dodatne (sekundarne) grupe korisnika na <code>GROUP1</code> i <code>GROUP2</code>	<code>useradd -m -G administrator,gameri anaanic</code>
<code>-c COMMENT</code>	Dodaje komentar (npr. puno ime i prezime)	<code>useradd -m -c "Ana Anić" anaanic</code>

Lozinku nije preporučljivo dodavati pomoću zastavice, već **naknadno pomoću** zasebne naredbe `passwd`:

```
→ sudo useradd -m -s /bin/bash -c "Ana Anić" anaanic # dodajemo korisnika "anaanic" s home direktorijem i zadanim shellom i punim imenom i prezimenom  
→ sudo passwd anaanic # dodajemo lozinku korisniku "anaanic"
```

```
Ubuntu Server [Running]  
lukablaskovic@ubuntu-server-vm:/home$ sudo useradd -m -s /bin/bash -c "Ana Anić" anaanic  
lukablaskovic@ubuntu-server-vm:/home$ ls  
anaanic lukablaskovic markomaric  
lukablaskovic@ubuntu-server-vm:/home$ sudo passwd anaanic  
New password:  
Retype new password:  
passwd: password updated successfully  
lukablaskovic@ubuntu-server-vm:/home$
```

Dodavanje lozinke korisniku `anaanic` koristeći `passwd` nakon uspješnog stvaranja korisnika naredbom `useradd -m -s`

- stvorili smo home direktorij `/home/anaanic`
- postavili smo zadani shell `/bin/bash`
- dodali smo komentar "Ana Anić"
- dodali smo lozinku korisniku `anaanic`, **naknadno**

Ako ostanemo u našem korisniku i pokušamo se prebaciti u home direktorij od `anaanic`, dobit ćemo grešku:

```
→ whoami # lukablaskovic  
→ cd /home/anaanic  
bash: cd: /home/anaanic: Permission denied
```

Neće raditi niti sa `sudo`, a niti nije moguće koristiti `sudo` s naredbom `cd`.

Dakle, nemamo privilegije za pristup *home* direktoriju `anaanic`, **jer je ona vlasnik tog direktorija**. Ipak, možemo ući unutra na dva načina:

- prijavimo se kao `anaanic` korisnik (`su - anaanic`)
- prijavimo se kao `root` korisnik (`sudo -i`) koji ima sve privilegije svijeta

Ako se prijavimo kao `anaanic`, automatski će nas prebaciti u `/home/anaanic`, međutim sad se ne možemo prebaciti u `/home/lukablaskovic` /`home/vaš_username`, niti se možemo prebaciti u `root` korisnika, čak i unosom točne lozinke - **jer nemamo sudo privilegije**.

```
→ whoami # anaanic
→ cd /home/lukablaskovic # Permission denied
→ sudo -i # Čak nakon unosa točne lozinke: anaanic is not in the sudoers file.
```

Možemo se odjaviti naredbom `exit`.

2.2 Naredba `usermod`

Do sad smo naučili da na Linuxu možemo imati više korisnika, od kojih su neki regularni korisnici, a neki su sustavni korisnici. Također, naučili smo da postoji i *superkorisnik* `root` koji ima sve privilegije.

Grupe (eng. *Group*) predstavljaju **skupine korisnika koji dijele iste privilegije**, preciznije: skupine korisnika koji imaju pristup istim resursima/datotekama u sustavu.

Možemo provjeriti koje grupe sadrži naš korisnik unosom naredbe `groups`:

```
→ whoami # lukablaskovic
→ groups # ispisuje sve grupe u kojima je korisnik član
# Ispisuje: lukablaskovic adm cdrom sudo dip plugdev lxd
```

Svaki korisnik može biti član:

- **jedne primarne grupu** (eng. primary group) i
- **jedne ili više dodatnih grupa** (eng. secondary groups).

Uočite da je naš korisnik član grupe `sudo`, što znači da može koristiti `sudo` privilegije - ima administratorske privilegije.

Provjerimo i za korisnika `anaanic`:

```
→ groups anaanic # ispisuje sve grupe u kojima je korisnik član
# Ispisuje: anaanic anaanic
```

`anaanic` korisnik je član samo svoje grupe `anaanic`, odnosno nije član nijedne druge dijeljene grupe.

Još jedna praktična naredba za provjeru grupe je naredba `id`.

Naredba `id` ispisuje informacije o korisniku, uključujući njegov **UID** (*User ID*) i **GID** (*Group ID*) za **svaku grupu kojoj pripada**.

```
→ whoami # lukablaskovic
→ id
# Ispisuje: uid=1000(lukablaskovic) gid=1000(lukablaskovic) groups=1000(lukablaskovic),
4(adm), 24(cdrom), 27(sudo), 30(dip), 46(plugdev), 101(lxd)
```

Čitamo:

- `uid=1000(lukablaskovic)` - User ID korisnika `lukablaskovic` je `1000`
- `gid=1000(lukablaskovic)` - Group ID - primarna grupa korisnika `lukablaskovic` je `1000`
- **ostale** (sekundarne) grupe su: `adm`, `cdrom`, `sudo`, `dip`, `plugdev`, `lxd`, a svaka ima svoj jedinstveni Group ID.

Napomena: privilegije koje dobivate u grupama možete najjednostavnije pronaći *Googlanjem*. Ove grupe su zadane grupe koje Ubuntu Server postavlja administratorskom korisniku, osim `sudo` prava za egzekuciju naredbi kao `root` korisnik, tu su i:

- `adm` - omogućava pristup log datotekama
- `plugdev` - omogućava pristup eksternim uređajima, poput USB uređaja
- `cdrom`, `dip`, `lxd` - omogućavaju pristup raznim uređajima i mrežnim sučeljima

Kako bismo korisniku dodali grupu, možemo koristiti naredbu `usermod`:

Naredba `usermod` se koristi za izmjenu postojećih korisnika u sustavu, odnosno njihovih postavki, uključujući dodavanje korisnika u grupu. Preciznije, **mi ovom naredbom uređujemo one konfiguracijske datoteke** `/etc/passwd`, `/etc/shadow`, `/etc/group` i `/etc/gshadow`, ovisno koju radnju izvodimo.

Sintaksa:

```
→ usermod [opcije/zastavice] <username>
```

Cilj je izmijeniti datoteku `/etc/group` i dodati korisnika u grupu. Koristimo sljedeće zastavice:

- `-a` - dodaje korisnika u grupu (append)
- `-G` - definira grupu u koju dodajemo korisnika

Na primjer, ako želimo dodati korisnika `anaanic` u grupu `sudo`, koristimo:

```
→ sudo usermod -a -G sudo anaanic
# ili spajanjem zastavica
→ sudo usermod -aG sudo anaanic
```

Provjeravamo grupe korisnika `anaanic`:

```
→ groups anaanic # vidimo "sudo" oznaku na kraju
```

Testirat ćemo da li korisnik `anaanic` može koristiti `sudo` privilegije:

```
→ su - anaanic # prebacujemo se u korisnika "anaanic"
→ sudo -i # pokušavamo se prebaciti u "root" korisnika
```

Ako je sve prošlo u redu, trebali bismo se prebaciti u `root` korisnika bez problema.

```
Ubuntu Server [Running]
lukablaskovic@ubuntu-server-vm:/home$ sudo usermod -a -G sudo anaanic
lukablaskovic@ubuntu-server-vm:/home$ groups anaanic
anaanic : anaanic sudo
lukablaskovic@ubuntu-server-vm:/home$ su - anaanic
Password:
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

anaanic@ubuntu-server-vm:~$ sudo -
[sudo] password for anaanic:
root@ubuntu-server-vm:~#
```

 Dodavanje `sudo` ovlasti korisniku `anaanic` pomoću naredbe `usermod`, prebacivanje u `root`

 **Napomena:** Pazite da ne zaboravite na zastavicu `-a` (append), jer ako ju ne navedemo, lista grupa će biti pregažena i korisnik će biti uklonjen iz svih ostalih grupa, osim iz one koju smo naveli.

usermod zastavica	Opis zastavice	Primjer
<code>-aG</code>	Dodaje korisniku sekundarne grupe (mora se koristiti s <code>-G</code>)	<code>usermod -aG sudo john</code>
<code>-G</code>	Postavlja dodatne (sekundarne) grupe (zamjenjuje postojeće ako se ne koristi <code>-a</code>)	<code>usermod -G dev,admin john</code>
<code>-g</code>	Mijenja primarnu grupu korisnika	<code>usermod -g developers john</code>
<code>-d</code>	Mijenja <i>home</i> direktorij	<code>usermod -d /newhome/john john</code>
<code>-m</code>	Premješta sadržaj u novi <i>home direktorij</i> (koristi se s <code>-d</code>)	<code>usermod -d /newhome/john -m john</code>
<code>-s</code>	Mijenja zadani <i>shell</i> korisnika	<code>usermod -s /bin/bash john</code>
<code>-l</code>	Mijenja korisničko ime	<code>usermod -l <newname> <oldname></code>
<code>-L</code>	Zaključava korisnički račun (onemogućuje prijavu)	<code>usermod -L john</code>
<code>-U</code>	Otključava korisnički račun	<code>usermod -U john</code>
<code>-e</code>	Postavlja datum isteka korisničkog računa	<code>usermod -e 2025-12-31 john</code>
<code>-c</code>	Mijenja komentar (obično puno ime korisnika)	<code>usermod -c "John Doe" john</code>

Primjer: Napisat ćemo bash skriptu koja očekuje jedan argument - korisničko ime, a zatim će zaključati korisnički račun - onemogućiti prijavu za tog korisnika.

```
→ whoami # lukablaskovic
→ cd /home/lukablaskovic
→ nano lock_user.sh
```

Prvo očekujemo jedan argument:

```
#!/bin/bash

if [ $# -ne 1 ]; then
    echo "Pogrešan unos, unesite: $0 <username>" # ispisujemo poruku upozorenja
    exit 1
fi
```

Provjeru postoji li korisnik možemo odraditi kroz naredbu `id`:

Ako korisni postoji, `id` će ispisati njegov `UID` i `GID` za sve grupe, inače će ispisati: "no such user".

Možemo pozvati naredbu s argumentom korisnika, a zatim provjeriti uspješnost (izlazni status) posljednje naredbe s `$?`:

```
id $1 # provjeravamo postoji li korisnik
if [ $? -ne 0 ]; then # ako korisnik ne postoji
    echo "Korisnik $1 ne postoji"
    exit 1
fi
```

Inače, ako korisnik postoji, možemo ga zaključati:

```
id $1 # provjeravamo postoji li korisnik
if [ $? -ne 0 ]; then # ako korisnik ne postoji
    echo "Korisnik $1 ne postoji"
    exit 1
fi

# zaključavamo korisnika
sudo usermod -L $1 # zaključavamo korisnika
```

Pozivamo skriptu s argumentom korisničkog imena:

```
→ chmod +x lock_user.sh
→ ./lock_user.sh anaanic
```

Ako je sve prošlo u redu, korisnik `anaanic` više se ne može prijaviti u sustav. Međutim, i dalje ćemo se moći prebaciti u njega kao `root` ili administrator korisnik. Otvorite novu terminal sesiju i pokušajte se prijaviti kao `anaanic` korisnik, ili pokušajte putem SSH izvana.

2.3 Naredba `userdel`

Naredba `userdel` se koristi za brisanje korisničkog računa iz sustava. Ova naredba također zahtijeva administratorske privilegije.

Sintaksa:

```
→ userdel [opcije/zastavice] <username>
```

- `<username>` - korisničko ime korisnika kojeg brišemo

Zastavice:

userdel zastavica	Opis zastavice	Primjer
<code>-r</code>	Briše korisnički račun i njegov home direktorij	<code>userdel -r anaanic</code>
<code>-f</code> ili <code>--force</code>	Brisanje "na silu", čak i ako je prijavljen ili ima aktivne procese	<code>userdel -f anaanic</code>

Primjer: Brišemo korisnika `anaanic` i njegov *home* direktorij:

```
→ sudo userdel -r anaanic
```

```
→ sudo userdel -f anaanic # brišemo korisnika, čak i ako je prijavljena i izvršava procese
```

Zadatak 3: Upravljanje korisnicima i grupama

Podignuli ste VM u cloud okruženju i prijavili se kao `root` korisnik. U vašoj organizaciji imate 2 junior developera na praksi i 1 senior developera. Njihova imena su:

- Senior developer: `crazy8Dev`, puno ime: `Domingo Gonzalez`
- Junor developer 1: `Cap_n_Cook`, puno ime: `Desi Rozi`
- Junor developer 2: `FlynnWebDev`, puno ime: `Flyn White`

Napravite korisničke račune za svakog od njih i dodijelite im sekundarnu grupu `devs`. Također, senior developeru dodijelite još jednu sekundarnu grupu `seniors` i `sudo` privilegije.

Dodajte lozinke za sve korisnike.

Kako su junior developeri na praksi, postavite datum isteka korisničkog računa na 1. listopada 2025. godine.

E sad, svake godine dobivate nove juniore i neda vam se ponovo raditi sve ispočetka. Prijavite se kao senior i u njegovom *home* direktoriju stvorite bash skriptu `create_junior.sh <username> <full_name> <password> <date>` koja će automatski dodati juniore u sustav pozivanjem svih odgovarajućih naredbi koje ste ručno upisivali.

3. Upravljanje servisima

Već smo se upoznali s osnovnim naredbama za upravljanje procesima, poput `ps`, `top`, `htop`, `kill`, `pkill` itd.

Servis (eng. service) je posebna vrsta procesa koji se pokreće u pozadini i obavlja određene funkcije ili zadatke. Servisi su obično **dugotrajni procesi** koji se pokreću prilikom pokretanja sustava i ostaju aktivni sve dok se sustav ne isključi.

Servisima upravlja *service manager* sustav, najčešće je to `systemd` koji je postavljen kao zadani servis manager na većini modernih Linux distribucija, uključujući Ubuntu Server.

`systemd` je također proces, a u njegovu aktivnost se možemo i uvjeriti preko `htop` alata ili `pidof` naredbe:

```
→ pidof systemd
```

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
1	root	20	0	21852	12128	8544	S	0.0	0.6	0:07.81	/sbin/init
287	root	19	-1	74940	18495	17344	S	0.0	0.9	0:04.16	/usr/lib/systemd/systemd-journald
340	root	RT	0	347M	26240	7296	S	0.0	1.3	0:05.79	/sbin/multipathd -d -s
351	root	20	0	347M	26240	7296	S	0.0	1.3	0:00.00	/sbin/multipathd -d -s
352	root	RT	0	347M	26240	7296	S	0.0	1.3	0:00.00	/sbin/multipathd -d -s
353	root	RT	0	347M	26240	7296	S	0.0	1.3	0:00.00	/sbin/multipathd -d -s
354	root	RT	0	347M	26240	7296	S	0.0	1.3	0:00.00	/sbin/multipathd -d -s
355	root	RT	0	347M	26240	7296	S	0.0	1.3	0:59.30	/sbin/multipathd -d -s
356	root	RT	0	347M	26240	7296	S	0.0	1.3	0:00.00	/sbin/multipathd -d -s
362	root	20	0	29260	7296	4480	S	0.0	0.4	0:01.66	/usr/lib/systemd/systemd-udevd
503	systemd-ne	20	0	18708	8448	7424	S	0.0	0.4	0:02.21	/usr/lib/systemd/systemd-networkd
512	systemd-re	20	0	21496	12160	9984	S	0.0	0.6	0:01.34	/usr/lib/systemd/systemd-resolved
518	systemd-ti	20	0	90760	6912	6016	S	0.0	0.3	0:03.61	/usr/lib/systemd/systemd-timesyncd
607	systemd-ti	20	0	90760	6912	6016	S	0.0	0.3	0:00.00	/usr/lib/systemd/systemd-timesyncd
648	messagebus	20	0	9968	4608	3712	S	0.0	0.2	0:00.65	@dbus-daemon --system --address=systemd
653	polkitd	20	0	301M	7424	6528	S	0.0	0.4	0:00.13	/usr/lib/polkit-1/polkitd --no-debug
662	root	20	0	17900	7808	6784	S	0.0	0.4	0:01.43	/usr/lib/systemd/systemd-logind
663	root	20	0	461M	12416	10368	S	0.0	0.6	0:00.91	/usr/libexec/udisks2/udisksd
677	root	20	0	461M	12416	10368	S	0.0	0.6	0:10.24	/usr/libexec/udisks2/udisksd
679	root	20	0	461M	12416	10368	S	0.0	0.6	0:00.09	/usr/libexec/udisks2/udisksd
682	root	20	0	461M	12416	10368	S	0.0	0.6	0:00.05	/usr/libexec/udisks2/udisksd
691	syslog	20	0	217M	4992	3712	S	0.0	0.2	0:00.54	/usr/sbin/rsyslogd -n -iNONE
700	root	20	0	105M	21632	12416	S	0.0	1.1	0:00.22	/usr/bin/python3 /usr/share/unattended-
705	polkitd	20	0	301M	7424	6528	S	0.0	0.4	0:15.90	/usr/lib/polkit-1/polkitd --no-debug

htop prikaz: Prikaz aktivnog `systemd` procesa.

U prošloj skripti naučili smo koristiti naredbu `systemctl`. Ova naredba koristi se za upravljanje servisima koji koriste `systemd` servisni manager.

Sintaksa:

```
→ systemctl [opcije/zastavice] <akcija> <servis>
```

- `<akcija>` - akcija koju želimo izvesti (npr. `start`, `stop`, `restart`, `status`, `enable`, `disable`)
- `<servis>` - naziv servisa koji želimo upravljati (npr. `apache2`, `ssh`, `mysql`, `nginx`, itd.)
- `[opcije/zastavice]` - dodatne opcije koje možemo koristiti (npr. `--now`, `--force`, `--quiet`, itd.)

Nećemo ulaziti u detalje svih mogućih opcija, već ćemo se fokusirati na najčešće korištene.

Primjer: Provjera statusa određenog servisa, npr `ssh`:

```
→ systemctl status ssh
```

Primjer: Provjera statusa svih aktivnih servisa:

```
→ systemctl status
```

Primjer: Pokretanje/zaustavljanje određenog servisa, npr `nginx`:

```
→ systemctl start nginx  
→ systemctl stop nginx  
→ systemctl restart nginx
```

Primjer: Provjera statusa određenog servisa, npr `ssh`:

```
→ systemctl status ssh
```

Primjer: Provjera koji servisi nisu uspješno pokrenuti prilikom podizanja sustava:

```
→ systemctl --failed  
  
# Ako želimo određene servise pokrenuti prilikom podizanja sustava, koristimo:  
→ systemctl enable <servis>  
  
# Ili pak onemogućiti:  
→ systemctl disable <servis>
```

Zadatak 4: Upravljanje servisima

Napišite bash skriptu koja prima naziv servisa kao argument (npr. `ssh`) i provjerava postoji li taj servis u sustavu naredbom `systemctl status`. Kako biste znali napisati `if` selekciju, provjerite što vraća varijabla `$?` kada servis postoji u sustavu, a što kada ne postoji.

Temeljem te selekcije, pokrenite određeni servis. Bash očekuje točno jedan argument - naziv servisa, a ako nije naveden, ispišite poruku upozorenja i izadite iz skripte.

Primjer upotrebe:

```
→ ./start_service.sh ssh  
  
Servis ssh je pokrenut  
  
→ ./start_service.sh apache2  
  
Servis apache2 ne postoji u sustavu
```

Zadaci za Vježbu 5

Objavim ih uskoro, ali to je to. Gradivo iz vježbi je gotovo! 