

گزارش دستورات ps,kill,top

گرد آورنده : رامین یزدانی

۱) دستور ps

ا. چیست؟

فرمان **ps** در لینوکس برای بررسی پردازش‌های در حال اجرا کاربرد دارد. می‌توانید درصد استفاده از رم، درصد استفاده از CPU، شناسه‌ی پردازش و نام دستور مربوط به پردازش و اطلاعات جزئی‌تر مثل نام کنسولی که کاربر لاگین کرده را به کمک فرمان **ps** پیدا کنید و در ادامه پردازش‌ها را مدیریت کنید.

ب. مدیریت پردازش‌ها در لینوکس با دستور ps

کرنل یا هسته، قلب تپنده‌ی سیستم عامل‌های شبیه **Unix** و تمام توزیعات لینوکس است. یکی از وظایف اصلی کرنل، استفاده از حافظه یا **RAM** و زمان پردازنده است. می‌بایست در هر لحظه این موارد بررسی و مدیریت شود و به پردازش‌های مختلف به شکل بهینه‌ای تخصیص پیدا کند. برخی پردازش‌ها **اولویت** بیشتری دارند و برخی کم‌اهمیت‌تر هستند و در هر لحظه نیز ممکن است اولویت‌ها تغییر کند. فرآیندی که به دلیلی اجرای آن طولانی شده یا به دلایل مختلف نرم‌افزاری، گیر کرده و نرم‌افزار پاسخگو نیست، ممکن است بخش زیادی از زمان پردازنده را بگیرد و مقدار زیادی از حافظه‌ی رم سیستم را اشغال کند که مطلوب نیست. در این مواقع کاربر حرفه‌ای تصمیم می‌گیرد که نرم‌افزار مربوطه را ببندد و نهایتاً پردازش را به اجبار متوقف کند.

شناسایی پردازشی که مشکل ساز شده یکی از نیازهای کاربران لینوکس است، درست مثل هر سیستم عامل دیگری و خوشبختانه برای این مهم، ابزارهای مختلفی در لینوکس پیش‌بینی شده است. دستور **ps** یکی از دستورات مفید در این زمینه است چرا که هم مشخصات پردازش‌ها شامل نام و شناسه و ارتباط با دیگر پردازش‌ها را نمایش می‌دهد و هم میزان استفاده از پردازنده توسط هر پردازش را ذکر می‌کند. این دستور چندین آپشن کاربردی نیز دارد.

ج. لیست کردن پردازش‌ها در Linux با ps

ساده‌ترین حالت کاربرد دستور **ps**، اجرا کردن آن در اپلیکیشن ترمینال است، بدون هیچ آپشنی!

```
Ps
ramin@Ramin:~$ ps
  PID TTY          TIME CMD
   9  tty1        00:00:00 bash
  22  tty1        00:00:00 ps
```

چهار ستون لیست موارد زیر است:

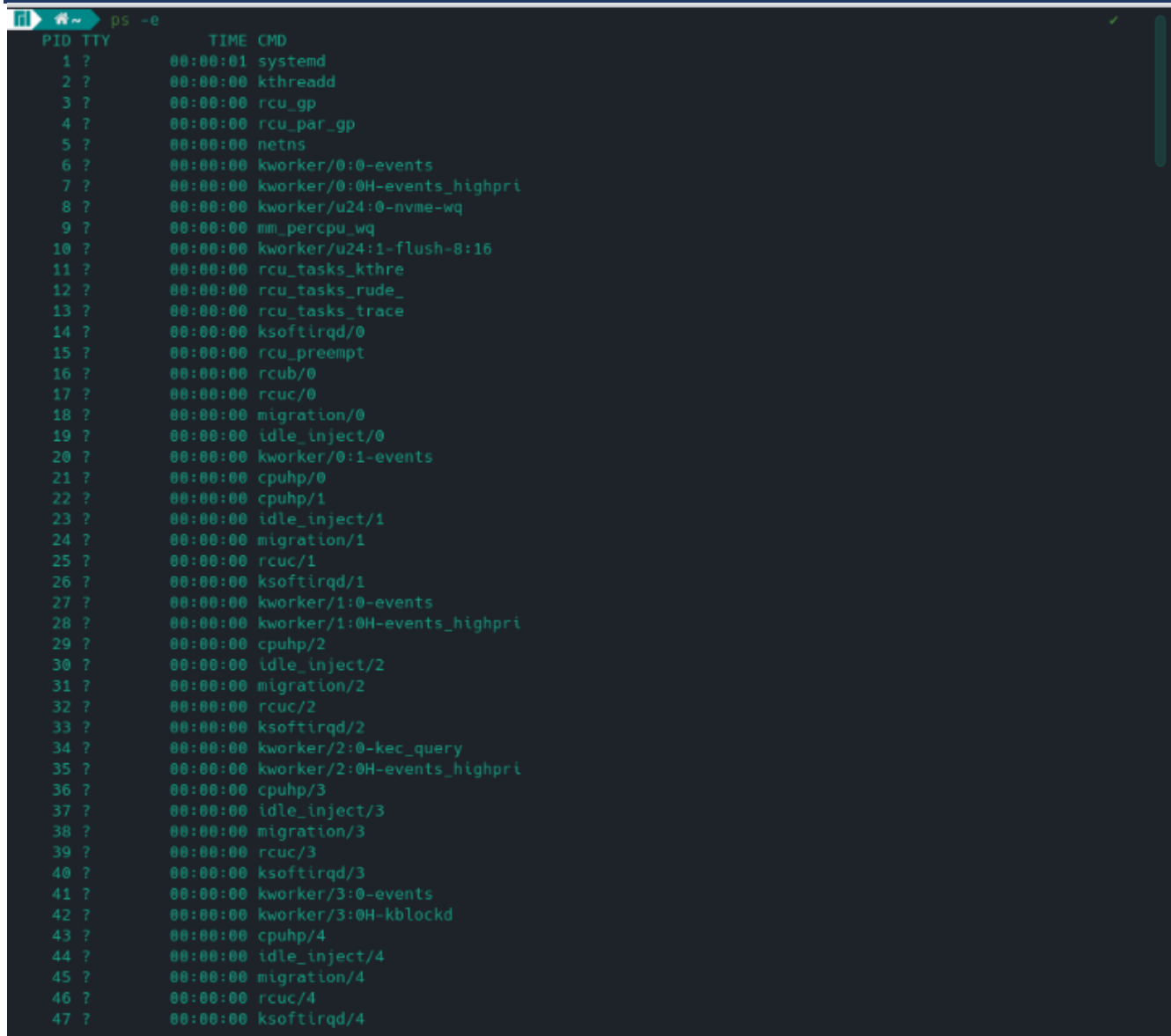
- **PID**: شناسه یا شماره‌ای که به هر پردازش اختصاص داده می‌شود.
- **TTY**: نام کنسولی که کاربرد در آن لاگین کرده است.
- **TIME**: مدت زمان پردازنده که به پردازش اختصاص پیدا کرده است که در واقع میزان استفاده از CPU است.
- **CMD**: نام دستوری که پردازش را اجرا کرده است.

د. لیست کردن پردازش‌های همه‌ی کاربران

برای چک کردن تمام پردازش‌ها، می‌توانید از دستور `ps` به همراه آپشن `-e` استفاده کنید که در این صورت پردازش‌های همه‌ی حساب‌های کاربری نیز لیست می‌شود. البته لیست پردازش‌ها در این صورت بسیار طولانی می‌شود چرا که در توزیعات مختلف لینوکس، تعداد زیادی حساب کاربری به صورت پیش‌فرض وجود دارد که صرفاً برای اجرا کردن پردازش‌های خاصی کاربرد دارند.

بد نیست پس از دستور `ps -e` از دستور `less` استفاده کنید تا چک کردن لیست ساده‌تر شود:

```
ps -e | less
```



همان‌طور که مشاهده می‌کنید در ستون `TTY`، علامت `?` ذکر شده که به این معنی است که برخی پردازش‌ها از طریق اجرای یک دستور در **Terminal** آغاز به کار نکرده‌اند.

دستور `less` توانایی اجرای کامند در یک صفحه ترمینال جدید را دارد. دارای اینترفیس فیچر ها برای کاربر به منظور جابه جایی و استفاده از کلید های میانبر در صفحه جدید است .

ه. نمایش نمودار درختی پردازش‌ها

می‌توانید از آپشن دیگری که مخفف `hierarchy` یا ساختار درختی است، برای نمایش پردازش‌ها به شکل درختی استفاده کنید. در واقع آپشن `-H` مشخص می‌کند که هر پردازش توسط کدام پردازش دیگری اجرا شده است.

```
ps -eH | less
```

```
1107 ?      00:00:00 krfcommd
1151 ?      00:00:01 kworker/u25:3+i915_flip
1 ?        00:00:01 systemd
358 ?      00:00:00 systemd-journal
359 ?      00:00:00 systemd-udev
602 ?      00:00:00 avahi-daemon
609 ?      00:00:00 avahi-daemon
603 ?      00:00:00 bluetoothd
604 ?      00:00:00 crond
605 ?      00:00:00 dbus-daemon
606 ?      00:00:00 polkitd
608 ?      00:00:00 systemd-logind
631 ?      00:00:00 NetworkManager
672 ?      00:00:00 cupsd
675 ?      00:00:00 ModemManager
676 ?      00:00:00 sddm
711 tty1    00:00:30 Xorg
745 ?      00:00:00 sddm-helper
760 ?      00:00:00 startplasma-x11
798 ?      00:00:00 plasma_session
819 ?      00:00:01 kded5
822 ?      00:00:35 kwin_x11
861 ?      00:00:00 kmsserver
891 ?      00:00:00 kaccess
896 ?      00:00:00 org_kde_powerde
897 ?      00:00:00 baloo_file
898 ?      00:00:18 plasmashell
1236 ?     00:00:06 dolphin
1321 ?     00:00:33 DesktopEditors
1325 ?     00:00:00 editors_helper
1361 ?     00:00:57 editors_helper
1376 ?     00:00:00 editors_helper
1338 ?     00:00:09 editors_helper
1358 ?     00:00:00 editors_helper
1727 ?     00:00:02 konsole
1745 pts/1   00:00:00 zsh
1815 pts/1   00:00:00 less
1817 pts/2   00:00:00 zsh
1927 pts/2   00:00:00 ps
1928 pts/2   00:00:00 less
899 ?      00:00:00 polkit-kde-auth
901 ?      00:00:00 xembedsniproxy
910 ?      00:00:00 gmenudbusmenupr
912 ?      00:00:00 kdeconnectd
914 ?      00:00:00 pamac-tray-plas
915 ?      00:00:00 msm_kde_notifie
918 ?      00:00:00 matray
923 ?      00:00:00 agent
```

ممکن است یک پردازش که از طریق ترمینال اجرا نشده و در واقع TTY آن علامت سوال است، چند پردازش دیگر را اجرا کند. در این صورت نام اجرا کننده عبارتی مثل `tty\` خواهد بود. برای بهتر شدن ظاهر لیست، می‌توانید از آپشن **forest**--استفاده کنید که خطوطی را به نمودار اضافه می‌کند:

```
ps -eH --forest | less
```

```
ramin@Ramin:~$ ps -eH --forest
PID TTY      TIME CMD
  1 ?          00:00:00 init
  8 tty1      00:00:00 init
  9 tty1      00:00:00 \_ bash
 68 tty1      00:00:00 \_ ps
```

آندرلاین یا - و بکاسلش یا علامت \ درک کردن نمودار را ساده‌تر می‌کند

و. لیست کردن پردازش‌ها با جستجوی نام توسط دستور **ps** و **grep**

می‌توانید دستور **ps** را در ترکیب با دستور **grep** استفاده کنید و با این روش نام یک پردازش را سرچ کنید. به عنوان مثال با اجرا کردن دستور زیر، هر پردازشی که در نام آن **rcu** موجود باشد لیست می‌شود:

```
ps -e | grep rcu
```

```

3 ?      00:00:00 rcu_gp
4 ?      00:00:00 rcu_par_gp
11 ?     00:00:00 rcu_tasks_kthre
12 ?     00:00:00 rcu_tasks_rude_
13 ?     00:00:00 rcu_tasks_trace
15 ?     00:00:00 rcu_preempt
16 ?     00:00:00 rcub/0
17 ?     00:00:00 rcuc/0
25 ?     00:00:00 rcuc/1
32 ?     00:00:00 rcuc/2
39 ?     00:00:00 rcuc/3
46 ?     00:00:00 rcuc/4
53 ?     00:00:00 rcuc/5
60 ?     00:00:00 rcuc/6
67 ?     00:00:00 rcuc/7
74 ?     00:00:00 rcuc/8
81 ?     00:00:00 rcuc/9
88 ?     00:00:00 rcuc/10
95 ?     00:00:00 rcuc/11
270 ?    00:00:00 kworker/1:2-rcu_gp

```

ز. دستور **ps** و ستون‌های متنوع اطلاعات پردازش‌ها

برای نمایش اطلاعات کامل‌تر می‌توانید ستون‌های اطلاعات پردازش‌ها را با آپشن **-f** یا **full-format** کامل کنید:

```
ps -ef | less
```

UID	PID	PPID	C	STIME	TTY	TIME	CMD
root	1	0	0	04:25	?	00:00:01	/sbin/init
root	2	0	0	04:25	?	00:00:00	[kthreadd]
root	3	2	0	04:25	?	00:00:00	[rcu_gp]
root	4	2	0	04:25	?	00:00:00	[rcu_par_gp]
root	5	2	0	04:25	?	00:00:00	[netns]
root	7	2	0	04:25	?	00:00:00	[kworker/0:0H-events_highpri]
root	9	2	0	04:25	?	00:00:00	[mm_percpu_wq]
root	11	2	0	04:25	?	00:00:00	[rcu_tasks_kthre]
root	12	2	0	04:25	?	00:00:00	[rcu_tasks_rude_]
root	13	2	0	04:25	?	00:00:00	[rcu_tasks_trace]
root	14	2	0	04:25	?	00:00:00	[ksoftirqd/0]
root	15	2	0	04:25	?	00:00:00	[rcu_preempt]
root	16	2	0	04:25	?	00:00:00	[rcub/0]
root	17	2	0	04:25	?	00:00:00	[rcuc/0]
root	18	2	0	04:25	?	00:00:00	[migration/0]
root	19	2	0	04:25	?	00:00:00	[idle_inject/0]
root	20	2	0	04:25	?	00:00:00	[kworker/0:1-inet_frag_wq]
root	21	2	0	04:25	?	00:00:00	[cpuhp/0]
root	22	2	0	04:25	?	00:00:00	[cpuhp/1]
root	23	2	0	04:25	?	00:00:00	[idle_inject/1]
root	24	2	0	04:25	?	00:00:00	[migration/1]
root	25	2	0	04:25	?	00:00:00	[rcuc/1]
root	26	2	0	04:25	?	00:00:00	[ksoftirqd/1]
root	28	2	0	04:25	?	00:00:00	[kworker/1:0H-events_highpri]
root	29	2	0	04:25	?	00:00:00	[cpuhp/2]
root	30	2	0	04:25	?	00:00:00	[idle_inject/2]
root	31	2	0	04:25	?	00:00:00	[migration/2]
root	32	2	0	04:25	?	00:00:00	[rcuc/2]
root	33	2	0	04:25	?	00:00:00	[ksoftirqd/2]
root	35	2	0	04:25	?	00:00:00	[kworker/2:0H-events_highpri]
root	36	2	0	04:25	?	00:00:00	[cpuhp/3]
root	37	2	0	04:25	?	00:00:00	[idle_inject/3]
root	38	2	0	04:25	?	00:00:00	[migration/3]
root	39	2	0	04:25	?	00:00:00	[rcuc/3]
root	40	2	0	04:25	?	00:00:00	[ksoftirqd/3]
root	42	2	0	04:25	?	00:00:00	[kworker/3:0H-kblockd]
root	43	2	0	04:25	?	00:00:00	[cpuhp/4]
root	44	2	0	04:25	?	00:00:00	[idle_inject/4]
root	45	2	0	04:25	?	00:00:00	[migration/4]

ستون‌های اطلاعات شامل موارد زیر است:

- **UID**: شناسه‌ی کاربری که مالک پردازش است.
 - **PID**: شناسه‌ی پردازش
 - **PPID**: شناسه‌ی پردازش بالادست و در واقع پردازش اجراکننده‌ی یک پردازش دیگر است.
 - **C**: تعداد فرزندان و به عبارتی تعداد پردازش‌های اجرا شده توسط یک پردازش
 - **STIME**: زمان شروع یا Start Time پردازش
 - **TTY**: نام کنسول کاربری که کاربر لاگین کرده است.
 - **TIME**: زمان پردازنده که به پردازش اختصاص پیدا کرده است.
 - **CMD**: دستوری که موجب اجرای پردازش شده است.
- دقت کنید که آپشن **F-** با **f-** متفاوت است و به معنی **extra full-format** یا فرمت کامل‌تر است.

ps -eF less										
UID	PID	PPID	C	SZ	RSS	PSR	STIME	TTY	TIME	CMD
root	1	0	0	41740	12156	11	04:25	?	00:00:01	/sbin/init
root	2	0	0	0	0	1	04:25	?	00:00:00	[kthreadd]
root	3	2	0	0	0	0	04:25	?	00:00:00	[rcu_gp]
root	4	2	0	0	0	0	04:25	?	00:00:00	[rcu_par_gp]
root	5	2	0	0	0	0	04:25	?	00:00:00	[netns]
root	7	2	0	0	0	0	04:25	?	00:00:00	[kworker/0:0H-events_highpri]
root	9	2	0	0	0	0	04:25	?	00:00:00	[mm_percpu_wq]
root	11	2	0	0	0	0	04:25	?	00:00:00	[rcu_tasks_kthre]
root	12	2	0	0	0	0	04:25	?	00:00:00	[rcu_tasks_rude_]
root	13	2	0	0	0	0	04:25	?	00:00:00	[rcu_tasks_trace]
root	14	2	0	0	0	0	04:25	?	00:00:00	[ksoftirqd/0]
root	15	2	0	0	0	0	04:25	?	00:00:00	[rcu_preempt]
root	16	2	0	0	0	0	04:25	?	00:00:00	[rcub/0]
root	17	2	0	0	0	0	04:25	?	00:00:00	[rcuc/0]
root	18	2	0	0	0	0	04:25	?	00:00:00	[migration/0]
root	19	2	0	0	0	0	04:25	?	00:00:00	[idle_inject/0]
root	20	2	0	0	0	0	04:25	?	00:00:00	[kworker/0:1-inet_frag_wq]
root	21	2	0	0	0	0	04:25	?	00:00:00	[cpuhp/0]
root	22	2	0	0	0	1	04:25	?	00:00:00	[cpuhp/1]
root	23	2	0	0	0	1	04:25	?	00:00:00	[idle_inject/1]
root	24	2	0	0	0	1	04:25	?	00:00:00	[migration/1]
root	25	2	0	0	0	1	04:25	?	00:00:00	[rcuc/1]
root	26	2	0	0	0	1	04:25	?	00:00:00	[ksoftirqd/1]
root	28	2	0	0	0	1	04:25	?	00:00:00	[kworker/1:0H-events_highpri]
root	29	2	0	0	0	2	04:25	?	00:00:00	[cpuhp/2]
root	30	2	0	0	0	2	04:25	?	00:00:00	[idle_inject/2]
root	31	2	0	0	0	2	04:25	?	00:00:00	[migration/2]
root	32	2	0	0	0	2	04:25	?	00:00:00	[rcuc/2]
root	33	2	0	0	0	2	04:25	?	00:00:00	[ksoftirqd/2]
root	35	2	0	0	0	2	04:25	?	00:00:00	[kworker/2:0H-events_highpri]
root	36	2	0	0	0	3	04:25	?	00:00:00	[cpuhp/3]
root	37	2	0	0	0	3	04:25	?	00:00:00	[idle_inject/3]
root	38	2	0	0	0	3	04:25	?	00:00:00	[migration/3]
root	39	2	0	0	0	3	04:25	?	00:00:00	[rcuc/3]
root	40	2	0	0	0	3	04:25	?	00:00:00	[ksoftirqd/3]
root	42	2	0	0	0	3	04:25	?	00:00:00	[kworker/3:0H-kblockd]
root	43	2	0	0	0	4	04:25	?	00:00:00	[cpuhp/4]
root	44	2	0	0	0	4	04:25	?	00:00:00	[idle_inject/4]
root	45	2	0	0	0	4	04:25	?	00:00:00	[migration/4]
root	46	2	0	0	0	4	04:25	?	00:00:00	[rcuc/4]
root	47	2	0	0	0	4	04:25	?	00:00:00	[ksoftirqd/4]
root	48	2	0	0	0	4	04:25	?	00:00:00	[kworker/4:0-mm_percpu_wq]
root	49	2	0	0	0	4	04:25	?	00:00:00	[kworker/4:0H-events_highpri]
root	50	2	0	0	0	5	04:25	?	00:00:00	[cpuhp/5]
root	51	2	0	0	0	5	04:25	?	00:00:00	[idle_inject/5]
root	52	2	0	0	0	5	04:25	?	00:00:00	[migration/5]
root	53	2	0	0	0	5	04:25	?	00:00:00	[rcuc/5]

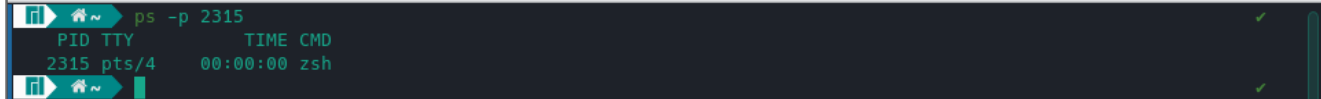
این بار لیست عریض‌تر و ستون‌ها بیشتر می‌شود و احتمالاً به اسکرول کردن افقی برای رویت کردن همه‌ی ستون‌ها کامل نیاز است! البته نیازی به استفاده از موس نیست، می‌توانید کلید **→** یا در واقع **فلش راست** را فشار دهید و بخش چپ لیست را مشاهده کنید.

- **SZ**: یا سایز و اندازه‌ی صفحات تصویر پردازش در RAM
- **RSS**: یا Resident set size که حافظه‌ی فیزیکی سوپ نشده‌ای است که پردازش استفاده می‌کند.
- **PSR**: یا پردازنده‌ای که پردازش به آن محول شده است.

ح. جستجوی پردازش‌ها با ID یا شناسه‌ی پردازش

می‌توانید پس از دستور ps از آپشن **p**-استفاده کنید و سپس شناسه‌ی پردازش را تایپ کنید تا اطلاعات مربوط به همان پردازش خاص لیست شود.

```
ps -p 2315
```



PID	TTY	TIME	CMD
2315	pts/4	00:00:00	zsh

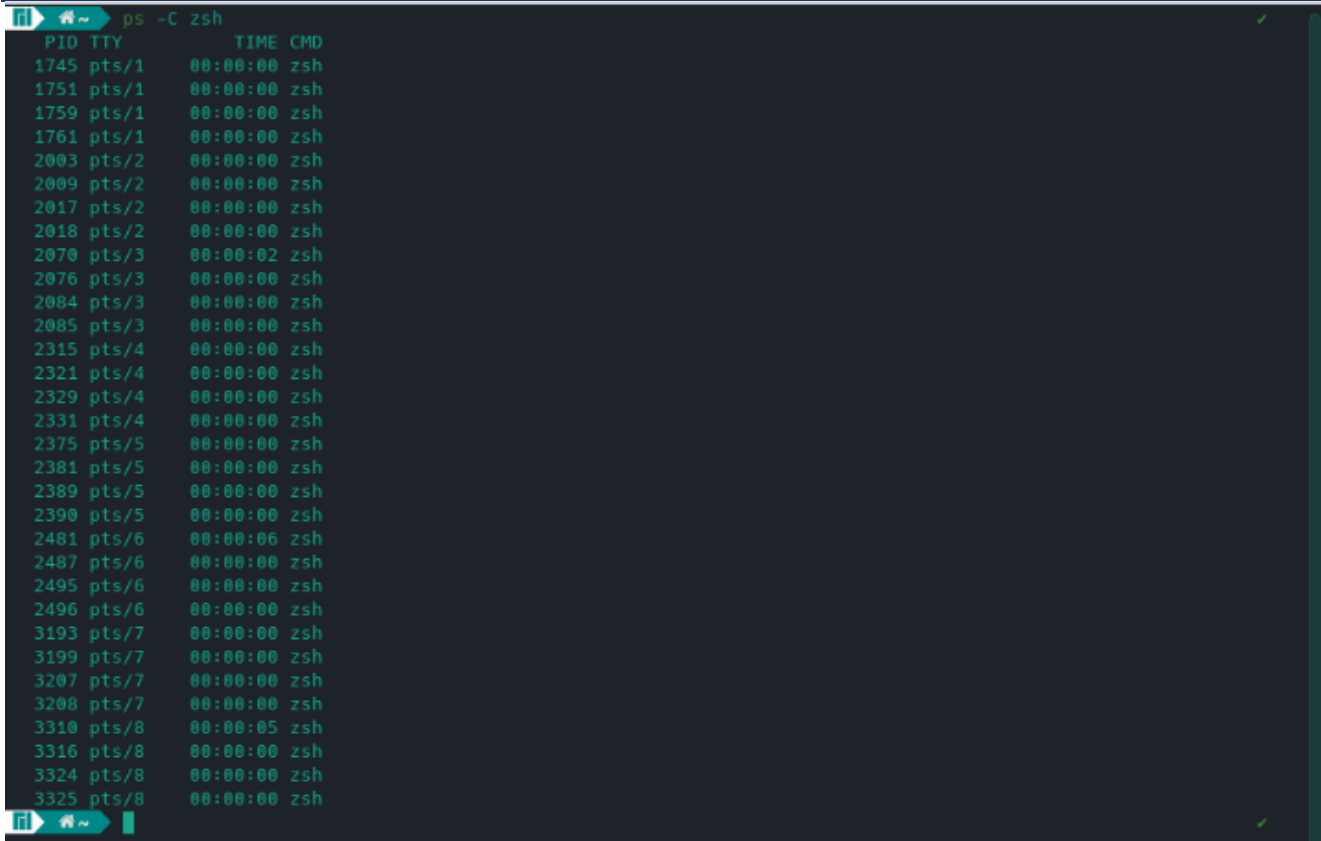
و خروجی دستور فوق ذکر مشخصات پردازشی با شناسه یا ID موردنظر است

اگر منظور شما جستجو کردن چند پردازش باشد هم می‌توانید شناسه‌ها را به ترتیب و با زدن اسپیس یا فاصله بینشان، پس از **ps -p** تایپ کنید.

ط. لیست کردن پردازش‌ها با دستور اجرای پردازش

برای سرچ کردن پردازش‌هایی که با دستور خاصی اجرا شده‌اند، می‌توانید از آپشن **C**- که مخفف Command است استفاده کنید. پس از این آپشن نام دستور مربوطه را تایپ کنید.

```
ps -C zsh
```

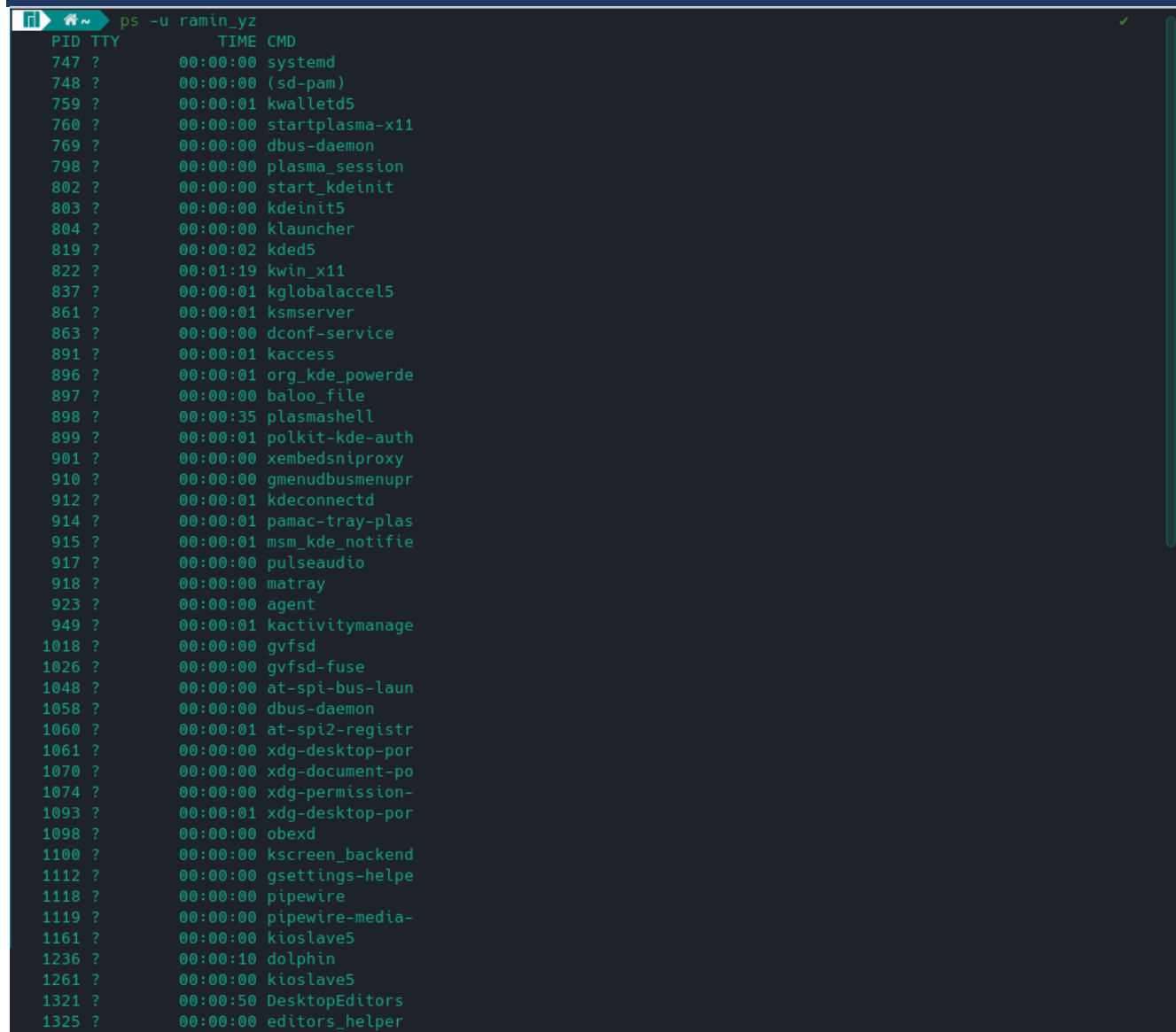


PID	TTY	TIME	CMD
1745	pts/1	00:00:00	zsh
1751	pts/1	00:00:00	zsh
1759	pts/1	00:00:00	zsh
1761	pts/1	00:00:00	zsh
2003	pts/2	00:00:00	zsh
2009	pts/2	00:00:00	zsh
2017	pts/2	00:00:00	zsh
2018	pts/2	00:00:00	zsh
2070	pts/3	00:00:02	zsh
2076	pts/3	00:00:00	zsh
2084	pts/3	00:00:00	zsh
2085	pts/3	00:00:00	zsh
2315	pts/4	00:00:00	zsh
2321	pts/4	00:00:00	zsh
2329	pts/4	00:00:00	zsh
2331	pts/4	00:00:00	zsh
2375	pts/5	00:00:00	zsh
2381	pts/5	00:00:00	zsh
2389	pts/5	00:00:00	zsh
2390	pts/5	00:00:00	zsh
2481	pts/6	00:00:06	zsh
2487	pts/6	00:00:00	zsh
2495	pts/6	00:00:00	zsh
2496	pts/6	00:00:00	zsh
3193	pts/7	00:00:00	zsh
3199	pts/7	00:00:00	zsh
3207	pts/7	00:00:00	zsh
3208	pts/7	00:00:00	zsh
3310	pts/8	00:00:05	zsh
3316	pts/8	00:00:00	zsh
3324	pts/8	00:00:00	zsh
3325	pts/8	00:00:00	zsh

ی. لیست پردازش‌های مربوط به کاربر خاص

آپشن بعدی `-u` یا `user list` است که پردازش‌های مربوط به یک حساب کاربری خاص را لیست می‌کند.

```
ps -u ramin_yz
```



PID	TTY	TIME	CMD
747	?	00:00:00	systemd
748	?	00:00:00	(sd-pam)
759	?	00:00:01	kwalletd5
760	?	00:00:00	startplasma-x11
769	?	00:00:00	dbus-daemon
798	?	00:00:00	plasma_session
802	?	00:00:00	start_kdeinit
803	?	00:00:00	kdeinit5
804	?	00:00:00	klauncher
819	?	00:00:02	kdcd5
822	?	00:01:19	kwin_x11
837	?	00:00:01	kglobalaccel5
861	?	00:00:01	ksmserver
863	?	00:00:00	dconf-service
891	?	00:00:01	kaccess
896	?	00:00:01	org_kde_powerde
897	?	00:00:00	baloo_file
898	?	00:00:35	plasmashell
899	?	00:00:01	polkit-kde-auth
901	?	00:00:00	xembedsniproxy
910	?	00:00:00	gmenudbusmenupr
912	?	00:00:01	kdeconnectd
914	?	00:00:01	pamac-tray-plas
915	?	00:00:01	msm_kde_notifie
917	?	00:00:00	pulseaudio
918	?	00:00:00	matray
923	?	00:00:00	agent
949	?	00:00:01	kactivitymanage
1018	?	00:00:00	gvfsd
1026	?	00:00:00	gvfsd-fuse
1048	?	00:00:00	at-spi-bus-laun
1058	?	00:00:00	dbus-daemon
1060	?	00:00:01	at-spi2-registr
1061	?	00:00:00	xdg-desktop-por
1070	?	00:00:00	xdg-document-po
1074	?	00:00:00	xdg-permission-
1093	?	00:00:01	xdg-desktop-por
1098	?	00:00:00	obexd
1100	?	00:00:00	kscreen_backend
1112	?	00:00:00	gsettings-helpe
1118	?	00:00:00	pipewire
1119	?	00:00:00	pipewire-media-
1161	?	00:00:00	kioslave5
1236	?	00:00:10	dolphin
1261	?	00:00:00	kioslave5
1321	?	00:00:50	DesktopEditors
1325	?	00:00:00	editors_helper

دستور فوق پردازش‌های حساب کاربری به اسم `ramin` را لیست می‌کند.

ک. لیست پردازش‌های ترمینال‌های مختلف

با آپشن `-t` می‌توانید پردازش‌هایی که به یک TTY را بررسی کنید. اگر پس از این دستور عددی تایپ نکنید، پردازش‌های مربوط به ترمینال فعلی لیست می‌شود.

```
ps -t
```

و در صورت ذکر عدد نیز نتیجه به صورت زیر است:

```
ps -t 1
```

```
ps -t 1
  PID TTY          TIME CMD
 1745 pts/1    00:00:00 zsh
 1751 pts/1    00:00:00 zsh
 1759 pts/1    00:00:00 zsh
 1761 pts/1    00:00:00 zsh
 1762 pts/1    00:00:00 gitstatusd
 1815 pts/1    00:00:00 less
```

ل. انتخاب ستون‌های لیست خروجی دستور ps

با استفاده از آپشن `-o` می‌توانید ستون‌های لیستی که با استفاده از دستور `ps` چاپ می‌شود را به صورت دلخواه انتخاب کنید. به عنوان مثال اگر فقط به دو ستون درصد استفاده از پردازنده یا `pcpu` و آرگومان‌های دستور یا `args` در خروجی نیاز داشته باشید، می‌توانید از دستور زیر استفاده کنید که در واقع بین نام این دو ستون، از ویرگول استفاده شده است:

```
ps -e -o pcpu,args | less
```

```
%CPU COMMAND
0.1 /sbin/init
0.0 [kthreadd]
0.0 [rcu_gp]
0.0 [rcu_par_gp]
0.0 [netns]
0.0 [kworker/0:0H-events_highpri]
0.0 [mm_percpu_wq]
0.0 [rcu_tasks_kthre]
0.0 [rcu_tasks_rude_]
0.0 [rcu_tasks_trace]
0.0 [ksoftirqd/0]
0.0 [rcu_preempt]
0.0 [rcub/0]
0.0 [rcuc/0]
0.0 [migration/0]
0.0 [idle_inject/0]
0.0 [cpuhp/0]
0.0 [cpuhp/1]
0.0 [idle_inject/1]
0.0 [migration/1]
0.0 [rcuc/1]
0.0 [ksoftirqd/1]
0.0 [kworker/1:0H-events_highpri]
0.0 [cpuhp/2]
0.0 [idle_inject/2]
0.0 [migration/2]
0.0 [rcuc/2]
0.0 [ksoftirqd/2]
0.0 [kworker/2:0H-events_highpri]
0.0 [cpuhp/3]
0.0 [idle_inject/3]
0.0 [migration/3]
0.0 [rcuc/3]
0.0 [ksoftirqd/3]
0.0 [kworker/3:0H-kblockd]
0.0 [cpuhp/4]
0.0 [idle_inject/4]
0.0 [migration/4]
0.0 [rcuc/4]
0.0 [ksoftirqd/4]
0.0 [kworker/4:0-mm_percpu_wq]
0.0 [kworker/4:0H-events_highpri]
0.0 [cpuhp/5]
```

مرتب‌سازی خروجی دستور ps و دیدن سنگین‌ترین پردازش‌ها

می‌توانید پردازش‌ها را به ترتیب اطلاعاتشان مرتب کنید. برای این موضوع از آپشن **--sort** استفاده کنید و پس از آن برای مرتب‌سازی از زیاد به کم، از خط فاصله و سپس نام ستون موردنظر را تایپ کنید. به عنوان مثال برای مرتب‌سازی بر حسب درصد استفاده از پردازنده، از **--sort -pcpu** استفاده کنید:

```
ps -e -o pcpu,args --sort -pcpu | less
```

```
%CPU COMMAND
19.2 /opt/onlyoffice/desktopeditors/editors_helper --type=zygote --no-sandbox --force-device-scale-factor=1 --log-fi
le=/home/ramin_yz/.local/share/onlyoffice/desktopeditors/data/cache/log.log --log-severity=disable --product-version
=Chrome/75 AscDesktopEditor/7.1.0 --lang=en-US
11.0 /usr/bin/kwin_x11
8.4 /usr/lib/Xorg -nolisten tcp -background none -seat seat0 vt1 -auth /var/run/sddm/{2f0e898d-28d3-40be-9a70-78900
157fff1} -noreset -displayfd 17
7.0 ./DesktopEditors /run/media/ramin_yz/windows_linux/دستورات ps و top و kill.docx
6.9 /bin/zsh
4.8 /usr/bin/plasmashell
3.1 /opt/onlyoffice/desktopeditors/editors_helper --type=gpu-process --field-trial-handle=14781086704786938409,3618
086415254803294,131072 --no-sandbox --disable-d3d11 --log-file=/home/ramin_yz/.local/share/onlyoffice/desktopeditors
/data/cache/log.log --log-severity=disable --product-version=Chrome/75 AscDesktopEditor/7.1.0 --lang=en-US --gpu-pre
ferences=KAAAAAAAAAAgAAAgAQAAAAAAAAAAAGAAAAAAAAAAAAIAAAAAAAAAAgAAAAAAAAAA --use-gl=swiftshader-webgl --log-file=/hom
e/ramin_yz/.local/share/onlyoffice/desktopeditors/data/cache/log.log --service-request-channel-token=483013327357582
1777
2.1 /bin/zsh
1.8 /usr/bin/konsole
1.5 /usr/bin/dolphin
0.5 [kworker/u25:3-i915_flip]
0.4 /bin/zsh
0.3 [irq/143-ELAN120]
0.3 /usr/bin/kded5
0.3 /usr/lib/kf5/kioslave5 /usr/lib/qt/plugins/kf5/kio/thumbnaill.so thumbnail local:/run/user/1000/plasmashellublZ
KK.8.slave-socket
0.2 [kworker/5:0-events_freezable]
0.2 /usr/lib/kactivitymanagerd
0.2 /usr/lib/kf5/kioslave5 /usr/lib/qt/plugins/kf5/kio/thumbnaill.so thumbnail local:/run/user/1000/dolphinEWQIAK.1
0.slave-socket
0.1 /sbin/init
0.1 /usr/bin/dbus-daemon --system --address=systemd: --nofork --nopidfile --systemd-activation --syslog-only
0.1 /usr/bin/kwalletd5 --pam-login 7 8
0.1 /usr/lib/kf5/klauncher --fd=9
0.1 /usr/bin/kglobalaccel5
0.1 /usr/bin/kmsserver
0.1 /usr/bin/kaccess
0.1 /usr/lib/org_kde_powerdevil
0.1 /usr/lib/polkit-kde-authentication-agent-1
0.1 /usr/bin/xembedsniproxy
0.1 /usr/bin/gmenubushmenuproxy
0.1 /usr/lib/kdeconnectd
0.1 /usr/lib/udisks2/udisksd
0.1 /usr/bin/pamac-tray-plasma
0.1 /usr/bin/msm_kde_notifier
0.1 /usr/lib/at-spi2-registryd --use-gnome-session
0.1 /usr/lib/xdg-desktop-portal-kde
0.1 /usr/lib/kf5/kscreen_backend_launcher
0.1 /usr/lib/baloorunner
```

همان‌طور که می‌بینید استفاده از **--sort** موجب شده که پردازش‌های سنگین‌تر در ابتدای لیست قرار بگیرند. اگر بخواهید تنها چند پردازش صدر لیست که به عنوان مثال بیشتر از پردازنده استفاده می‌کنند را مشاهده کنید، می‌توانید دستور ps را با دستور **head** ترکیب کنید. به عنوان مثال با اجرای فرمان زیر، ۱۰ پردازشی که بیشتری درصد استفاده از پردازنده را به خود اختصاص داده‌اند، لیست می‌شوند.

```
ps -e -o pcpu,args --sort -pcpu | head -10
```

```
%CPU COMMAND
19.0 /opt/onlyoffice/desktopeditors/editors_helper --type=zygote --no-sandbox --force-device-scale-factor=1 --log-file=/home/ramin_yz/.local/share/onlyoffice/desktopeditors/data/cache/log.log --log-severity=disable --product-version=Chrome/75 AscDesktopEditor/7.1.0 --lang=en-US
11.1 /usr/bin/kwin_x11
8.7 /bin/zsh
8.4 /usr/lib/Xorg -nolisten tcp -background none -seat seat0 vt1 -auth /var/run/sddm/{2f0e898d-28d3-40be-9a70-78900157fff1} -noreset -displayfd 17
6.8 ./DesktopEditors /run/media/ramin_yz/windows_linux/دستورات ps و top و kill.docx
4.9 /usr/bin/plasmashell
3.1 /opt/onlyoffice/desktopeditors/editors_helper --type=gpu-process --field-trial-handle=14781086704786938409,3618086415254803294,131072 --no-sandbox --disable-d3d11 --log-file=/home/ramin_yz/.local/share/onlyoffice/desktopeditors/data/cache/log.log --log-severity=disable --product-version=Chrome/75 AscDesktopEditor/7.1.0 --lang=en-US --gpu-preferences=KAAAAAAAAAAGAAAGAQAAAAAAAAAGAAAAAAAAAAAAIAAAAAAAAAAGAAAAAAAAA --use-gl=swiftshader-webgl --log-file=/home/ramin_yz/.local/share/onlyoffice/desktopeditors/data/cache/log.log --service-request-channel-token=4830133273575821777
1.9 /bin/zsh
1.8 /usr/bin/konsole
```

ستون مفید دیگر در لیست پردازش‌ها، ستون **pmem** یا Percent of Memory به معنی درصد استفاده از حافظه است. زمانی که سیستم لینوکسی کند شده است، ممکن است با کمبود رم یا پردازنده مواجه باشید و لذا این ستون نیز بسیار مفید و کاربردی است. با اجرا کردن فرمان زیر، ۱۰ پردازش که بیشتر از پردازنده استفاده کرده‌اند و در درجه‌ی بعدی بر حسب کمترین درصد استفاده از حافظه مرتب شده‌اند، ذکر می‌شود:

```
ps -e -o pcpu,pmem,args --sort -pcpu,pmem | head -10
```

```
%CPU %MEM COMMAND
18.9 2.2 /opt/onlyoffice/desktopeditors/editors_helper --type=zygote --no-sandbox --force-device-scale-factor=1 --log-file=/home/ramin_yz/.local/share/onlyoffice/desktopeditors/data/cache/log.log --log-severity=disable --product-version=Chrome/75 AscDesktopEditor/7.1.0 --lang=en-US
10.9 0.9 /usr/bin/kwin_x11
8.4 0.8 /usr/lib/Xorg -nolisten tcp -background none -seat seat0 vt1 -auth /var/run/sddm/{2f0e898d-28d3-40be-9a70-78900157fff1} -noreset -displayfd 17
6.7 1.8 ./DesktopEditors /run/media/ramin_yz/windows_linux/دستورات ps و top و kill.docx
4.8 1.8 /usr/bin/plasmashell
3.2 0.0 /bin/zsh
3.1 0.6 /opt/onlyoffice/desktopeditors/editors_helper --type=gpu-process --field-trial-handle=14781086704786938409,3618086415254803294,131072 --no-sandbox --disable-d3d11 --log-file=/home/ramin_yz/.local/share/onlyoffice/desktopeditors/data/cache/log.log --log-severity=disable --product-version=Chrome/75 AscDesktopEditor/7.1.0 --lang=en-US --gpu-preferences=KAAAAAAAAAAGAAAGAQAAAAAAAAAGAAAAAAAAAAAAIAAAAAAAAAAGAAAAAAAAA --use-gl=swiftshader-webgl --log-file=/home/ramin_yz/.local/share/onlyoffice/desktopeditors/data/cache/log.log --service-request-channel-token=4830133273575821777
1.7 0.0 /bin/zsh
1.7 0.8 /usr/bin/konsole
```

و خروجی دستور فوق که در آن ترتیب بر اساس درصد استفاده از پردازنده و سپس کمترین استفاده از رم است برای شناسایی ساده‌تر پردازش‌ها در لینوکس، بهتر است همواره ستون شناسه‌ی پردازش را نیز چک کنید. لذا با اضافه کردن یک **ویرگول** و نام ستون شناسه که **pid** است، یک ستون دیگر به لیست اضافه می‌کنیم:

```
ps -e -o pid,pcpu,pmem,args --sort -pcpu,pmem | head -10
```

```
PID %CPU %MEM COMMAND
1361 18.8 2.2 /opt/onlyoffice/desktopeditors/editors_helper --type=zygote --no-sandbox --force-device-scale-factor=1 --log-file=/home/ramin_yz/.local/share/onlyoffice/desktopeditors/data/cache/log.log --log-severity=disable --product-version=Chrome/75 AscDesktopEditor/7.1.0 --lang=en-US
822 11.0 0.9 /usr/bin/kwin_x11
711 8.4 0.8 /usr/lib/Xorg -nolisten tcp -background none -seat seat0 vt1 -auth /var/run/sddm/{2f0e898d-28d3-40be-9a70-78900157fff1} -noreset -displayfd 17
1321 6.6 1.8 ./DesktopEditors /run/media/ramin_yz/windows_linux/دستورات ps و top و kill.docx
898 4.8 1.8 /usr/bin/plasmashell
1338 3.1 0.6 /opt/onlyoffice/desktopeditors/editors_helper --type=gpu-process --field-trial-handle=14781086704786938409,3618086415254803294,131072 --no-sandbox --disable-d3d11 --log-file=/home/ramin_yz/.local/share/onlyoffice/desktopeditors/data/cache/log.log --log-severity=disable --product-version=Chrome/75 AscDesktopEditor/7.1.0 --lang=en-US --gpu-preferences=KAAAAAAAAAAGAAAGAQAAAAAAAAAGAAAAAAAAAAAAIAAAAAAAAAAGAAAAAAAAA --use-gl=swiftshader-webgl --log-file=/home/ramin_yz/.local/share/onlyoffice/desktopeditors/data/cache/log.log --service-request-channel-token=4830133273575821777
3310 2.3 0.0 /bin/zsh
1727 1.8 0.8 /usr/bin/konsole
1236 1.6 0.8 /usr/bin/dolphin
```

اکنون شناسایی پردازش‌ها ساده‌تر است

۲) دستور top

ا. چیست؟

با استفاده از دستور top در لینوکس فرآیندهای اجرا شده بر روی سیستم لینوکس خود را در دست بگیرید. دستور top در لینوکس آمار مفیدی در مورد منابع سیستم ارائه می دهد. ما می توانیم از آن برای مشاهده استفاده از CPU و حافظه در کنار اطلاعات فرآیند سرویس های در حال اجرا استفاده کنیم. همچنین می توانید فرآیندهای زامبی را با استفاده از top پیدا کنید. بنابراین ، تسلط بر دستور top برای هرکسی که با لینوکس کار می کند ، ضروری است.

ب. فرمان top چگونه کار می کند؟

به طور پیش فرض ، top لیستی از فرآیندهای در حال اجرا را در کنار معیارهای استاندارد CPU نشان می دهد. می توانید قسمت اول خروجی را به عنوان داشبورد در نظر بگیرید. قسمت پایین لیست فرآیند را نشان می دهد و یک نمایش زمان واقعی از تمام فرآیندهای در حال اجرا را ارائه می دهد. داشبورد از پنج خط تشکیل شده است که هر کدام دارای معیارهایی هستند.

- خط اول اطلاعات کوتاهی در مورد سیستم ، مانند زمان به روز ، میانگین بار و تعداد کاربرانی که در حال حاضر وارد شده اند را نشان می دهد.
- وظایف در خط دوم نشان داده شده است.
- خط سوم بار CPU را نشان می دهد
- دو خط زیر نشان دهنده میزان استفاده از حافظه است.

توجه داشته باشید که دستوراتی که هنگام اجرای top مشخص می کنید به حروف کوچک و بزرگ حساس هستند. به عنوان مثال ، کلیدهای n و N هر دو عملیات متفاوتی را انجام می دهند.

ج. نمایش کلیه فرآیندهای در حال اجرا

هنگامی که بدون هیچ گونه استدلال استفاده می شود ، فرمان top لیستی از تمام فرآیندهای در حال اجرا را نشان می دهد.

top

```
top - 04:41:38 up 16 min, 10 users, load average: 0.96, 0.82, 0.62
Tasks: 333 total, 1 running, 332 sleeping, 0 stopped, 0 zombie
%Cpu(s): 1.4 us, 0.9 sy, 0.0 ni, 97.6 id, 0.0 wa, 0.1 hi, 0.0 si, 0.0 st
MiB Mem : 15851.4 total, 12753.9 free, 1448.5 used, 1649.0 buff/cache
MiB Swap: 8192.0 total, 8192.0 free, 0.0 used. 13806.9 avail Mem

  PID USER      PR  NI   VIRT   RES   SHR  S  %CPU  %MEM     TIME+ COMMAND
 1361 ramin_yz  20   0  13.1g 366152 102652 S   15.0   2.3   2:57.50 editors_helper
 1321 ramin_yz  20   0 3588968 296704 153568 S    3.3   1.8   1:02.44 DesktopEditors
 1338 ramin_yz  20   0 668228 112144  75968 S    2.7   0.7   0:29.97 editors_helper
 1727 ramin_yz  20   0 2011356 134096 104088 S    1.3   0.8   0:13.87 konsole
  711 root      20   0  25.9g 137144  94916 S    1.0   0.8   1:23.31 Xorg
  822 ramin_yz  20   0 3151568 149216 107768 S    1.0   0.9   1:48.24 kwin_x11
  195 root      20   0      0      0      0 I    0.3   0.0   0:00.66 kworker/u24:6-events_unbound
 3553 ramin_yz  20   0  10976   4220  3360 R    0.3   0.0   0:00.01 top
    1 root      20   0 166960 12156  9032 S    0.0   0.1   0:01.58 systemd
    2 root      20   0      0      0      0 S    0.0   0.0   0:00.01 kthreadd
    3 root      0 -20      0      0      0 I    0.0   0.0   0:00.00 rcu_gp
    4 root      0 -20      0      0      0 I    0.0   0.0   0:00.00 rcu_par_gp
    5 root      0 -20      0      0      0 I    0.0   0.0   0:00.00 netns
    7 root      0 -20      0      0      0 I    0.0   0.0   0:00.00 kworker/0:0H-events_highpri
    9 root      0 -20      0      0      0 I    0.0   0.0   0:00.00 mm_percpu_wq
   11 root      20   0      0      0      0 S    0.0   0.0   0:00.00 rcu_tasks_kthre
   12 root      20   0      0      0      0 S    0.0   0.0   0:00.00 rcu_tasks_rude_
   13 root      20   0      0      0      0 S    0.0   0.0   0:00.00 rcu_tasks_trace
   14 root      20   0      0      0      0 S    0.0   0.0   0:00.10 ksoftirqd/0
   15 root      -2   0      0      0      0 I    0.0   0.0   0:00.67 rcu_preempt
   16 root      -2   0      0      0      0 S    0.0   0.0   0:00.00 rcub/0
   17 root      -2   0      0      0      0 S    0.0   0.0   0:00.00 rcuc/0
   18 root      rt   0      0      0      0 S    0.0   0.0   0:00.00 migration/0
   19 root     -51   0      0      0      0 S    0.0   0.0   0:00.00 idle_inject/0
```

می توانید با استفاده از کلیدهای بالا ، پایین ، PageUp و PageDown روی صفحه کلید جابجها شوید.

د. مرتب سازی فرآیندهای لینوکس بر اساس PID

می توانید لیست فرآیند را با شناسه برنامه یا PID آنها مرتب کنید. برای مرتب سازی فرایندها بر اساس PID ، کلید N را هنگام top فشار دهید.

N

ه. مرتب سازی فرایندها بر اساس حافظه و استفاده از CPU

خروجی بالا به طور پیش فرض لیست فرآیند را بر اساس میزان استفاده از CPU مرتب می کند. می توانید لیست را با استفاده از حافظه با استفاده از کلید M روی صفحه کلید مرتب کنید P را وارد کنید مرتب سازی بر اساس CPU می شود.

M

```
top - 04:41:50 up 16 min, 10 users, load average: 1.19, 0.88, 0.64
Tasks: 334 total, 1 running, 333 sleeping, 0 stopped, 0 zombie
%Cpu(s): 6.9 us, 2.8 sy, 0.0 ni, 89.2 id, 0.0 wa, 0.9 hi, 0.2 si, 0.0 st
MiB Mem : 15851.4 total, 12753.0 free, 1443.6 used, 1654.8 buff/cache
MiB Swap: 8192.0 total, 8192.0 free, 0.0 used, 13806.4 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1361	ramin_yz	20	0	13.1g	372396	107404	S	22.5	2.3	3:00.05	editors_helper
898	ramin_yz	20	0	5947352	305396	152740	S	12.3	1.9	0:50.23	plasmashell
1321	ramin_yz	20	0	3588968	296704	153568	S	5.8	1.8	1:03.12	DesktopEditors
822	ramin_yz	20	0	3151728	149216	107768	S	33.3	0.9	1:52.54	kwin_x11
711	root	20	0	25.9g	137144	94916	S	20.7	0.8	1:26.20	Xorg
1236	ramin_yz	20	0	2168220	135268	106668	S	2.5	0.8	0:16.30	dolphin
1727	ramin_yz	20	0	2011356	134188	104088	S	2.5	0.8	0:14.02	konsole
1338	ramin_yz	20	0	673348	115572	79396	S	4.7	0.7	0:30.50	editors_helper
1376	ramin_yz	20	0	1821268	108772	77836	S	0.0	0.7	0:00.45	editors_helper
1611	ramin_yz	20	0	2315408	106276	81416	S	0.0	0.7	0:02.67	kioslave5
1578	ramin_yz	20	0	2304356	103180	78676	S	0.4	0.6	0:03.93	kioslave5
918	ramin_yz	20	0	1274500	81576	61272	S	0.4	0.5	0:00.60	matray
819	ramin_yz	20	0	1499672	74996	62028	S	0.4	0.5	0:03.13	kded5
1358	ramin_yz	20	0	849808	65336	54472	S	0.0	0.4	0:00.10	editors_helper
912	ramin_yz	20	0	370400	57960	49364	S	0.4	0.4	0:01.54	kdeconnectd
915	ramin_yz	20	0	470976	55516	46764	S	0.4	0.3	0:01.51	msm_kde_notifie
1335	ramin_yz	20	0	333660	55468	45312	S	0.0	0.3	0:00.03	editors_helper

P

```
top - 04:42:27 up 17 min, 10 users, load average: 1.11, 0.91, 0.66
Tasks: 332 total, 1 running, 331 sleeping, 0 stopped, 0 zombie
%Cpu(s): 1.4 us, 0.7 sy, 0.0 ni, 97.8 id, 0.0 wa, 0.1 hi, 0.0 si, 0.0 st
MiB Mem : 15851.4 total, 12746.8 free, 1452.0 used, 1652.6 buff/cache
MiB Swap: 8192.0 total, 8192.0 free, 0.0 used, 13805.8 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1361	ramin_yz	20	0	13.1g	367084	103020	S	14.6	2.3	3:05.44	editors_helper
1321	ramin_yz	20	0	3588968	296704	153568	S	3.3	1.8	1:04.54	DesktopEditors
1338	ramin_yz	20	0	668228	112072	75896	S	2.3	0.7	0:31.39	editors_helper
711	root	20	0	25.9g	137848	94916	S	1.0	0.8	1:30.70	Xorg
822	ramin_yz	20	0	3151472	149204	107836	S	1.0	0.9	1:59.48	kwin_x11
1727	ramin_yz	20	0	2011512	134188	104088	S	0.7	0.8	0:14.81	konsole
195	root	20	0	0	0	0	I	0.3	0.0	0:00.73	kworker/u24:6-flush-8:16
348	root	0	-20	0	0	0	I	0.3	0.0	0:00.02	kworker/3:1H-kblockd
1060	ramin_yz	20	0	161016	6768	5884	S	0.3	0.0	0:01.49	at-spi2-registr
2972	root	20	0	0	0	0	I	0.3	0.0	0:00.16	kworker/u24:1-events_unbound
3553	ramin_yz	20	0	10976	4220	3360	R	0.3	0.0	0:01.07	top
1	root	20	0	166960	12156	9032	S	0.0	0.1	0:01.60	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.01	kthreadd
3	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_gp
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_par_gp
5	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	netns
7	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0H-events_highpri
9	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	mm_percpu_wq
11	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_tasks_kthre
12	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_tasks_rude_
13	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_tasks_trace
14	root	20	0	0	0	0	S	0.0	0.0	0:00.11	ksoftirqd/0
15	root	-2	0	0	0	0	I	0.0	0.0	0:00.71	rcu_preempt
16	root	-2	0	0	0	0	S	0.0	0.0	0:00.00	rcub/0
17	root	-2	0	0	0	0	S	0.0	0.0	0:00.00	rcuc/0
18	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	migration/0
19	root	-51	0	0	0	0	S	0.0	0.0	0:00.00	idle_inject/0
21	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/0
22	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/1

و. مرتب سازی فرایندها بر اساس زمان اجرا

اگر می خواهید مدت زمان اجرای فرایندها را در دستگاه خود ببایید ، کلیدهای M و T را فشار دهید.

T

```
top - 04:42:44 up 17 min, 10 users, load average: 1.07, 0.91, 0.67
Tasks: 333 total, 1 running, 332 sleeping, 0 stopped, 0 zombie
%Cpu(s): 3.1 us, 1.0 sy, 0.0 ni, 95.6 id, 0.0 wa, 0.2 hi, 0.1 si, 0.0 st
MiB Mem : 15851.4 total, 12716.3 free, 1464.0 used, 1671.1 buff/cache
MiB Swap: 8192.0 total, 8192.0 free, 0.0 used, 13775.8 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1361	ramin_yz	20	0	13.1g	373660	107508	S	14.3	2.3	3:08.10	editors_helper
822	ramin_yz	20	0	3153520	149220	107836	S	15.1	0.9	2:02.32	kwin_x11
711	root	20	0	25.9g	137848	94916	S	9.2	0.8	1:32.80	Xorg
1321	ramin_yz	20	0	3588968	296704	153568	S	4.2	1.8	1:05.24	DesktopEditors
898	ramin_yz	20	0	5955160	308116	152928	S	4.2	1.9	0:55.17	plasmashell
1338	ramin_yz	20	0	673348	115456	79280	S	2.5	0.7	0:31.91	editors_helper
1236	ramin_yz	20	0	2168500	135700	106668	S	4.2	0.8	0:18.54	dolphin
1727	ramin_yz	20	0	2011512	134188	104088	S	5.0	0.8	0:14.96	konsole
2481	ramin_yz	20	0	13040	9236	4800	S	0.0	0.1	0:06.90	zsh
1151	root	0	-20	0	0	0	I	0.8	0.0	0:05.71	kworker/u25:3-i915_flip
1578	ramin_yz	20	0	2304356	103180	78676	S	0.0	0.6	0:04.54	kioslave5
455	root	-51	0	0	0	0	S	0.0	0.0	0:04.02	irq/143-ELAN120
819	ramin_yz	20	0	1499672	74996	62028	S	0.8	0.5	0:03.26	kded5
949	ramin_yz	20	0	544832	39800	34208	S	0.0	0.2	0:03.17	kactivitymanage
1611	ramin_yz	20	0	2315408	106276	81416	S	5.0	0.7	0:02.98	kioslave5
55	root	20	0	0	0	0	I	0.0	0.0	0:02.35	kworker/5:0-events
3310	ramin_yz	20	0	12672	8824	4612	S	0.0	0.1	0:02.32	zsh
2070	ramin_yz	20	0	12520	8796	4628	S	0.0	0.1	0:02.27	zsh
605	dbus	20	0	10032	6312	4256	S	0.0	0.0	0:02.09	dbus-daemon

ز. نمایش فرایندهای در حال اجرا برای کاربران خاص

ما می توانیم لیستی از تمام فرایندهای در حال اجرا را که متعلق به یک کاربر خاص است مشاهده کنیم. وقتی در دستور Top هستیم ، u را فشار دهید و سپس نام کاربری را وارد کرده و Enter را فشار دهید . برای انجام این کار می توانید از گزینه -u به همراه نام کاربری استفاده کنید. یا زدن u و سرچ کردن نام کاربری

top -u root

```
top - 04:43:37 up 18 min, 10 users, load average: 0.77, 0.85, 0.66
Tasks: 334 total, 1 running, 333 sleeping, 0 stopped, 0 zombie
%Cpu(s): 2.9 us, 1.0 sy, 0.0 ni, 96.0 id, 0.0 wa, 0.1 hi, 0.1 si, 0.0 st
MiB Mem : 15851.4 total, 12730.0 free, 1466.8 used, 1654.6 buff/cache
MiB Swap: 8192.0 total, 8192.0 free, 0.0 used, 13790.6 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1	root	20	0	166960	12156	9032	S	0.0	0.1	0:01.61	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.01	kthreadd
3	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_gp
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_par_gp
5	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	netns
7	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0H-events_highpri
9	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	mm_percpu_wq
11	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_tasks_kthre
12	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_tasks_rude_

top -u raminyz

```
top - 04:43:14 up 18 min, 10 users, load average: 0.81, 0.86, 0.65
Tasks: 333 total, 1 running, 332 sleeping, 0 stopped, 0 zombie
%Cpu(s): 2.4 us, 0.6 sy, 0.0 ni, 96.9 id, 0.0 wa, 0.1 hi, 0.1 si, 0.0 st
MiB Mem : 15851.4 total, 12718.5 free, 1470.3 used, 1662.7 buff/cache
MiB Swap: 8192.0 total, 8192.0 free, 0.0 used, 13778.6 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1361	ramin_yz	20	0	13.1g	366724	104032	S	14.6	2.3	3:14.66	editors_helper
1321	ramin_yz	20	0	3588968	296704	153568	S	3.6	1.8	1:06.84	DesktopEditors
3745	ramin_yz	20	0	286564	48452	42476	S	3.3	0.3	0:00.10	spectacle
1338	ramin_yz	20	0	669044	113708	77532	S	2.3	0.7	0:33.24	editors_helper
822	ramin_yz	20	0	3151536	149224	107836	S	1.0	0.9	2:04.54	kwin_x11
1727	ramin_yz	20	0	2011356	134320	104152	S	1.0	0.8	0:15.30	konsole
3737	ramin_yz	20	0	10976	4216	3356	R	1.0	0.0	0:00.04	top
769	ramin_yz	20	0	9308	5308	4064	S	0.3	0.0	0:00.97	dbus-daemon

ح. فرآیندهای فعال را برجسته کنید

اگر کلید Z را در top وارد کنید ، تمام فرآیندهای فعال لینوکس را برجسته می کند . این حرکت فرآیندهای فعال را ساده تر می کند.

ط. تغییر دوره بازه top

به طور پیش فرض ، top خروجی خود را هر سه ثانیه تازه می کند. با این حال ، شما می توانید به راحتی این مقدار را با فشار دادن d و سپس مقدار مورد نیاز ، روی مقدار سفارشی تنظیم کنید.

top -d [number]

```
top - 04:43:59 up 18 min, 10 users, load average: 0.77, 0.85, 0.66
Tasks: 333 total, 2 running, 331 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.5 us, 1.0 sy, 0.0 ni, 98.5 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 15851.4 total, 12726.6 free, 1463.7 used, 1661.1 buff/cache
MiB Swap: 8192.0 total, 8192.0 free, 0.0 used. 13787.7 avail Mem
Change delay from 10.0 to
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1361	ramin_yz	20	0	13.1g	363220	101524	S	13.3	2.2	3:21.84	editors_helper
837	ramin_yz	20	0	282852	46096	39056	S	6.7	0.3	0:01.97	kglobalaccel5
1321	ramin_yz	20	0	3588968	296880	153724	S	6.7	1.8	1:08.68	DesktopEditors
1	root	20	0	166960	12156	9032	S	0.0	0.1	0:01.61	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.01	kthreadd
3	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_gp
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_par_gp
5	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	netns
7	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0H-events_highpri
9	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	mm_percpu_wq
11	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_tasks_kthre
12	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_tasks_rude_
13	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_tasks_trace
14	root	20	0	0	0	0	S	0.0	0.0	0:00.13	ksoftirqd/0
15	root	-2	0	0	0	0	R	0.0	0.0	0:00.78	rcu_preempt
16	root	-2	0	0	0	0	S	0.0	0.0	0:00.00	rcub/0
17	root	-2	0	0	0	0	S	0.0	0.0	0:00.00	rcuc/0
18	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	migration/0

یا بعد از top زدن کلید d و مشخص کردن زمان delay

ی. تغییر اولویت فرآیند

شما می توانید اولویت یک فرآیند لینوکس را با تعیین مقدار Renice سفارشی در بالا تغییر دهید r. را به همراه PID فرآیند تایپ کرده و سپس مقدار Renice جدید آن را وارد کنید.

ک. نمایش فرآیندهای بیکار با استفاده از دستور top لینوکس

ما می توانیم با فشار دادن کلید i لیستی از تمام فرآیندهای بیکار را مشاهده کنیم.

i

```
top - 04:44:22 up 19 min, 10 users, load average: 0.79, 0.84, 0.66
Tasks: 333 total, 1 running, 332 sleeping, 0 stopped, 0 zombie
%Cpu(s): 1.5 us, 0.4 sy, 0.0 ni, 97.9 id, 0.0 wa, 0.1 hi, 0.1 si, 0.0 st
MiB Mem : 15851.4 total, 12722.8 free, 1463.9 used, 1664.8 buff/cache
MiB Swap: 8192.0 total, 8192.0 free, 0.0 used. 13784.3 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1361	ramin_yz	20	0	13.1g	364896	102500	S	15.7	2.2	3:25.78	editors_helper
1321	ramin_yz	20	0	3588968	296880	153724	S	2.9	1.8	1:09.77	DesktopEditors
1338	ramin_yz	20	0	668228	112236	76060	S	2.9	0.7	0:35.23	editors_helper
711	root	20	0	25.9g	139936	94916	S	2.0	0.9	1:42.19	Xorg
759	ramin_yz	20	0	283212	45160	39508	S	1.0	0.3	0:01.37	kwalletd5
915	ramin_yz	20	0	470976	55516	46764	S	1.0	0.3	0:01.73	msm_kde_notifie
1060	ramin_yz	20	0	161016	6768	5884	S	1.0	0.0	0:01.64	at-spi2-registr
3809	ramin_yz	20	0	10976	4308	3444	R	1.0	0.0	0:00.07	top

ل. کشتن یک فرآیند لینوکس توسط PID

دستور **top** در لینوکس به ما اجازه می دهد تا یک فرآیند در حال اجرا را مستقیماً از طریق رابط از بین ببریم. برای از بین بردن یک فرآیند ، عبارت **k** را وارد کرده و PID آن فرآیند را وارد کنید. هنگام برخورد با فرایندهای زامبی مفید خواهد بود.

حالت های اصلی فرایندها در لینوکس عبارتند از:

- خواب (S) : فرایندهایی که منتظر نوبت اجرای آنها هستند.
- در حال اجرا (R) : فرایندهایی که در حال اجرا هستند.
- انتظار (D) : فرایندهای منتظر تکمیل عملیات ورود / خروج.
- زامبی (Z) : فرایندهایی که پایان یافته اند اما همچنان در جدول فرآیند ظاهر می شوند. می توانند ناشی از خطاهای برنامه نویسی باشند و می توانند نشانه ای از یک سیستم کند یا مشکل آفرین باشند.

فرآیند زامبی فرآیندی است که هرگز سیگنالی از فرآیند والد که آن را ایجاد کرده دریافت نکرده است ، فرآیند کودک فرآیندی است که منشأ آن فرآیند سطح بالاتری است که به عنوان فرآیند والد شناخته می شود و مسئول ارسال سیگنال ها به فرایندهای کودک است. این نشان می دهد که عمر آنها به پایان رسیده است.

۳) دستور های kill و pkill و killall

ا. چیست؟

زمانی که برنامه‌ای در یکی از توزیعات لینوکس هنگ می‌کند و به درستی کار نمی‌کند، احتمالاً بستن آن با روش گرافیکی و ساده، غیرممکن می‌شود. در این صورت باید مثل ویندوز و مک‌اواس، پردازش مربوط به آن را به صورت اجباری بست. در غیر این صورت ممکن است درصد بالایی از توان پردازشی CPU را به خود اختصاص بدهد و مقدار زیادی RAM اشغال کند. برای بستن اجباری پردازش‌ها در لینوکس، چند فرمان ساده مثل kill و pkill و killall وجود دارد که هر یک کاربرد خاص خود را دارد.

ب. پردازش در لینوکس چیست؟

زمانی که برنامه‌ای را اجرا می‌کنید و در محیط گرافیکی آن مشغول کار می‌شوید، در حقیقت تعداد زیادی پردازش و دستور در حال اجراست. در حالت کلی پردازش‌ها در سیستم عامل Linux و همین‌طور ویندوز و مک‌اواس، به دو دسته تقسیم می‌شوند:

ا) پردازش‌های پیش‌زمینه که کاربر آنها را اجرا کرده است. البته ممکن است اجرا کردن یک اپلیکیشن گرافیکی، موجب آغاز به کار این پردازش‌ها شده باشد و مستقیماً دستور یا پردازشی اجرا نشده باشد.

ب) پردازش‌های پس‌زمینه یا بک‌گراند که به صورت خودکار شروع به کار می‌کنند و کاربر مستقیماً یا به صورت غیرمستقیم آنها را اجرا نکرده است. این پردازش‌ها برای فراهم کردن سرویس‌هایی در سیستم عامل، در پشت صحنه مشغول به کار می‌شوند.

زمانی که برنامه‌ای کرش و هنگ می‌کند، قابلیت بستن آن با کلیک روی دکمه‌ی بستن وجود ندارد و به همین جهت است که گاهی باید از طریق ترمینال، دستورات مربوط به بستن اجباری را اجرا کرد تا برنامه بالاچار بسته شود.

یک نکته‌ی مهم:

ممکن است به اپلیکیشن هنگ کرده نیازی نداشته باشید و بخواهید پس از ری‌استارت کردن کامپیوتر، آن را از نو اجرا کنید اما به هر حال بستن اجباری مفید است چرا که نمی‌گذارد پردازش‌های مربوط به یک برنامه، شدیداً از پردازنده و رم، کارت گرافیک و حتی هارددیسک سیستم شما استفاده کنند و فرآیندی را به شکل ناقص، مرتباً تکرار کنند.

منظور از Killing یا کشتن در لینوکس، بستن اجباری پردازش‌ها است که با چند دستور ساده و کاربردی انجام می‌شود. در ادامه با شیوه‌ی استفاده از kill و pkill و killall آشنا می‌شویم.

ج. بستن اجباری پردازش با دستور kill در لینوکس

برای استفاده از دستور kill، می‌بایست پس از این دستور، شناسه‌ی پردازش و به عبارت دیگر PID آن را تایپ کنید. برای چک کردن لیست پردازش‌ها و نمایش شناسه‌ی هر یک از پردازش‌ها، می‌توانید از دستور ps استفاده کنید. (قبلاً توضیح داده شده)

فرمان ps سوییچی به اسم -e دارد که موجب لیست شدن تمام پردازش‌ها می‌شود. برای راحت‌تر چک کردن لیست پردازش‌ها، می‌توانید پس از این دستور، دستور less را اضافه کنید. بنابراین فرمان زیر را در اپلیکیشن ترمینال لینوکس تایپ کنید:

```
ps -e | less
```

```
PID TTY          TIME CMD
 1 ?            00:00:01 systemd
 2 ?            00:00:00 kthreadd
 3 ?            00:00:00 rcu_gp
 4 ?            00:00:00 rcu_par_gp
 5 ?            00:00:00 netns
 7 ?            00:00:00 kworker/0:0H-events_highpri
 9 ?            00:00:00 mm_percpu_wq
11 ?            00:00:00 rcu_tasks_kthre
12 ?            00:00:00 rcu_tasks_rude_
13 ?            00:00:00 rcu_tasks_trace
14 ?            00:00:00 ksoftirqd/0
15 ?            00:00:00 rcu_preempt
16 ?            00:00:00 rcub/0
17 ?            00:00:00 rcuc/0
18 ?            00:00:00 migration/0
19 ?            00:00:00 idle_inject/0
21 ?            00:00:00 cpuhp/0
22 ?            00:00:00 cpuhp/1
23 ?            00:00:00 idle_inject/1
24 ?            00:00:00 migration/1
25 ?            00:00:00 rcuc/1
26 ?            00:00:00 ksoftirqd/1
28 ?            00:00:00 kworker/1:0H-events_highpri
29 ?            00:00:00 cpuhp/2
30 ?            00:00:00 idle_inject/2
31 ?            00:00:00 migration/2
32 ?            00:00:00 rcuc/2
33 ?            00:00:00 ksoftirqd/2
35 ?            00:00:00 kworker/2:0H-events_highpri
36 ?            00:00:00 cpuhp/3
37 ?            00:00:00 idle_inject/3
38 ?            00:00:00 migration/3
39 ?            00:00:00 rcuc/3
40 ?            00:00:00 ksoftirqd/3
42 ?            00:00:00 kworker/3:0H-kblockd
43 ?            00:00:00 cpuhp/4
44 ?            00:00:00 idle_inject/4
45 ?            00:00:00 migration/4
46 ?            00:00:00 rcuc/4
47 ?            00:00:00 ksoftirqd/4
48 ?            00:00:00 kworker/4:0-events
49 ?            00:00:00 kworker/4:0H-events_highpri
50 ?            00:00:00 cpuhp/5
51 ?            00:00:00 idle_inject/5
52 ?            00:00:00 migration/5
53 ?            00:00:00 rcuc/5
54 ?            00:00:00 ksoftirqd/5
```

با زدن کلید **Enter**، لیست پردازش‌ها نمایان می‌شود. در ستون اول، **PID** یا **Process ID** که شناسه‌ی پردازش است، ذکر می‌شود. اکنون می‌توانید با فشار دادن کلید **/** صفحات بعدی را ببینید و با زدن **?** صفحات قبلی را مشاهده کنید.

روش سریع‌تر برای یافتن شناسه‌ی پردازش، این است که با دستور **grep** در لیستی که با فرمان **ps -e** ساخته شده، جستجو انجام بدهید. کافی است پس از این دستور، از **|** و سپس دستور **grep** و در نهایت بخشی از نام برنامه یا پردازش استفاده کنید. به عنوان مثال برای یافتن برنامه‌ای که کلمه‌ی **zsh** در نام آن به کار رفته، از دستور زیر استفاده کنید:

```
ps -e | grep zsh
```

```
1745 pts/1 00:00:00 zsh
1751 pts/1 00:00:00 zsh
1759 pts/1 00:00:00 zsh
1761 pts/1 00:00:00 zsh
2003 pts/2 00:00:00 zsh
2009 pts/2 00:00:00 zsh
2017 pts/2 00:00:00 zsh
2018 pts/2 00:00:00 zsh
2070 pts/3 00:00:02 zsh
2076 pts/3 00:00:00 zsh
2084 pts/3 00:00:00 zsh
2085 pts/3 00:00:00 zsh
2315 pts/4 00:00:00 zsh
2321 pts/4 00:00:00 zsh
2329 pts/4 00:00:00 zsh
2331 pts/4 00:00:00 zsh
2375 pts/5 00:00:00 zsh
2381 pts/5 00:00:00 zsh
2389 pts/5 00:00:00 zsh
2390 pts/5 00:00:00 zsh
2481 pts/6 00:00:06 zsh
2487 pts/6 00:00:00 zsh
2495 pts/6 00:00:00 zsh
2496 pts/6 00:00:00 zsh
3193 pts/7 00:00:00 zsh
3199 pts/7 00:00:00 zsh
3207 pts/7 00:00:00 zsh
3208 pts/7 00:00:00 zsh
3310 pts/8 00:00:05 zsh
3316 pts/8 00:00:00 zsh
3324 pts/8 00:00:00 zsh
3325 pts/8 00:00:00 zsh
```

پس از یافتن شناسه‌ی پردازش، کافی است فرمان `kill` و شماره‌ی پردازش را وارد کنید و `Enter` را فشار دهید تا پردازش موردنظر بسته شود.

```
kill [pid number]
```

```
1745 pts/1 00:00:00 zsh
1751 pts/1 00:00:00 zsh
1759 pts/1 00:00:00 zsh
1761 pts/1 00:00:00 zsh
2003 pts/2 00:00:00 zsh
2009 pts/2 00:00:00 zsh
2017 pts/2 00:00:00 zsh
2018 pts/2 00:00:00 zsh
2070 pts/3 00:00:02 zsh
2076 pts/3 00:00:00 zsh
2084 pts/3 00:00:00 zsh
2085 pts/3 00:00:00 zsh
2315 pts/4 00:00:00 zsh
2321 pts/4 00:00:00 zsh
2329 pts/4 00:00:00 zsh
2331 pts/4 00:00:00 zsh
2375 pts/5 00:00:00 zsh
2381 pts/5 00:00:00 zsh
2389 pts/5 00:00:00 zsh
2390 pts/5 00:00:00 zsh
2481 pts/6 00:00:06 zsh
2487 pts/6 00:00:00 zsh
2495 pts/6 00:00:00 zsh
2496 pts/6 00:00:00 zsh
3193 pts/7 00:00:00 zsh
3199 pts/7 00:00:00 zsh
3207 pts/7 00:00:00 zsh
3208 pts/7 00:00:00 zsh
3310 pts/8 00:00:06 zsh
3316 pts/8 00:00:00 zsh
3324 pts/8 00:00:00 zsh
3325 pts/8 00:00:00 zsh
```

```
kill 1745
```

این فرمان نتیجه‌ای به عنوان خروجی نمایش نمی‌دهد و در واقع بدون سروصدا، پردازش را می‌بندد.

د. کار با دستور pkill در Linux

فرمان **pkill** هم برای بستن اجباری پردازش‌ها به کار می‌رود اما نیازی به شناسه یا PID نیست بلکه می‌بایست نام پردازش را پس از این دستور تایپ کنید. البته باید نام پردازش را صحیح تایپ کنید تا پردازشی با نام مشابه، به اشتباه بسته نشود! نکته‌ی مهم این است که **pkill** هر پردازشی که بخشی از نام آن را تایپ کرده باشید هم می‌بندد! برای جلوگیری از این رفتار، می‌بایست نام‌ها را جستجو کنید و نام کامل را تایپ کنید. برای اطمینان بیشتر می‌توانید از دستور **pgrep** برای جستجو کردن نام پردازش‌ها و برگرداندن شناسه استفاده کنید. طبعاً اگر نام پردازشی را اشتباه تایپ کرده باشید، نتیجه‌ی جستجو این اشتباه تایپی را روشن می‌کند. به عنوان مثال فرض کنید که می‌خواهید پردازشی با عبارت **pts/6** در نام آن را به صورت اجباری ببندید. با اجرا کردن فرمان **ps** و استفاده از **grep**، می‌توان مطمئن شد که تنها یک پردازش با عبارت **pts/6** در نام آن وجود دارد و البته نام کامل پردازش نیز ذکر می‌شود. در این مثال، **ramin_yz** نام حساب کاربری است:

```
ps -u ramin_yz | grep pts/6
```

```
ps -u ramin_yz | grep pts/6
2481 pts/6    00:00:06 zsh
2487 pts/6    00:00:00 zsh
2495 pts/6    00:00:00 zsh
2496 pts/6    00:00:00 zsh
2498 pts/6    00:00:00 gitstatusd
3166 pts/6    00:00:00 less
```

اکنون دستور زیر را اجرا کنید تا روشن شود که تنها یک پردازش با کلمه‌ی **pts/6** در نام آن وجود دارد و سپس با دستور **pkill** پردازش موردبحث را ببندید:

```
pgrep gitstatusd
```

```
pgrep gitstatusd
1762
2020
2087
2332
2392
2498
3210
3327
```

```
pkill gitstatusd
```

```
pkill gitstatusd
gitstatus_query_p9k:print:68: write error: broken pipe
```

همان‌طور که در ابتدا اشاره کردیم، **pkill** هر پردازشی که بخشی از نام آن با عبارت وارد شده یکی باشد را می‌بندد. به عنوان مثال: برای بستن چند پردازش در یک مرحله هم می‌توانید از دستور **pkill** استفاده کنید. به عنوان مثال مرورگر **firefox** را در نظر بگیرید، این مرورگر چند پردازش اجرا می‌کند. با اجرا کردن دستور زیر، این پردازش‌ها لیست می‌شود:

```
pgrep firefox
```

```
pgrep firefox
4901
```

و با اجرای دستور بعدی، همه‌ی پردازش‌ها بسته می‌شوند:

```
pkill firefox
```

```
pgrep firefox
4901
pkill firefox
```

در نهایت می‌توانید دستور اول را مجدداً اجرا کنید و بررسی کنید که آیا پردازشی با کلمه‌ی **firefox** در حال اجراست یا خیر:

```
pgrep firefox
```

اما حالت دیگر این است که چند پردازش تشابه اسمی دارند و نمی‌خواهید همه را ببندید. در این صورت می‌توانید فرمان **pgrep** را با سوییچ **-f** اجرا کنید تا تفاوت‌ها روشن شود. به عنوان مثال اگر بخواهید فرمان **ping** خاصی را ببندید، می‌توانید نام کامل آن را بین دو **دابل کوتیشن** که همان " است، قرار بدهید و در مورد این پردازش خاص جستجو کنید و در ادامه با فرمان **pkill** آن را ببندید:

```
pgrep -f "ping 192.168.4.22"
pkill -f "ping 192.168.4.22"
```

۵. بستن همه‌ی پردازش‌های یکسان با **killall**

استفاده از **kill** و **pkill** زمانی خوب است که پردازش‌های موردنظر چند مورد خاص و غیرتکراری باشد. اگر پردازشی داشته باشید که در زمان‌های مختلف به صورت مجزا و چندین مرتبه اجرا شده، برای بستن همه‌ی موارد آن بهتر است از دستور **killall** استفاده کنید. البته فرمان **killall** در برخی توزیعات لینوکس، برای بستن همه‌ی پردازش‌ها کاربرد دارد و اجرا کردن آن **خطرناک** است. در سیستم عامل **Solaris** و **OpenIndiana** با اجرای دستور **killall**، تمام پردازش‌های مربوط به حساب کاربری فعلی بسته می‌شود! اگر در حساب **روت** این دستور را اجرا کنید، فرمان **sudo killall** اجرا شده و موجب **ریبوت** شدن کامپیوتر لینوکسی می‌شود. دستور **killall** مشابه **pkill** است با این تفاوت که لازم است نام پردازش را به صورت کامل وارد کنید. به عنوان مثال دستور زیر موجب بسته شدن پردازشی به اسم **zsh** نمی‌شود در حالیکه مورد بعدی این کار را انجام می‌دهد:

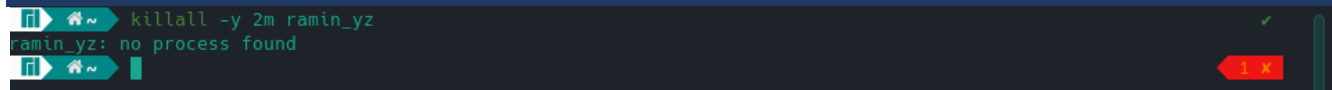
```
killall firefox
```

همان‌طور که در تصویر زیر مشاهده می‌کنید، نتیجه‌ی اجرای دستور اول، پیام **no process found** یا پیدا نکردن پردازش است. یکی از سوییچ‌های مفید **killall**، سوییچ **-y** یا **Younger Than** است که پردازش‌هایی که عمرشان کمتر از مقدار مشخص شده باشد را می‌بندد. عمر را می‌توانید با وارد کردن حروف **S** و **m** و **M** و **H** و غیره وارد کنید.

- **s** مخفف seconds یا ثانیه‌ها
- **m** مخفف minutes یا دقائق
- **h** مخفف hours یا ساعت‌ها
- **d** مخفف days یا روزها
- **w** مخفف weeks یا هفته‌ها
- **M** مخفف months یا ماه‌ها
- **y** مخفف years یا سال‌ها

به عنوان مثال برای بستن چند فرآیند پردازشی به اسم **ramin** که برخی اخیراً اجرا شده‌اند، می‌توانید از دستور زیر استفاده کنید: که تمام موارد مربوط به ۲ دقیقه‌ی اخیر را به صورت اجباری می‌بندد و موارد قدیمی را باز نگه می‌دارد:

```
killall -y 2m ramin_yz
```



سوییچ مشابه، سوییچ **-o** یا **Older Than** است که پردازش‌های قدیمی‌تر از بازه‌ی زمانی مشخص شده را می‌بندد.

```
killall -o 1d zsh
```