Operating Systems
سیستمهای عامل

مجموعه اسلایدهای شماره ۴

دکتر خانمیرزا

h.khanmirza@kntu.ac.ir

دانشکده کامپیوتر

دانشگاه صنعتی خواجه نصیرالدین طوسی

فرآیندها
Processes

فرآیند

- فرآیند یک abstraction برای اجرای یک برنامه است.
- در حالت سنتی (یا در تعاریف قدیمی) فرآیند عموما دارای یک خط اجراست (یا یک ریسمان اجرایی دارد).
- در هر فرآیند دارای دو بخش اصلی است:
  - بخش کد و اجرای ترتیبی و پشت سر هم آن و حالت ذخیره شده اجرایی
  - بخش منابع محافظت شده
    - منابع مورد نیاز فرآیند که شامل فضای آدرس و منابع مورد نیاز (نظیر فایلهای باز شده) است
    - این فضا همانند یک container بوده و اطلاعات آن از سایر فرآیندها جداست و توسط سیستم عامل از دسترسیهای غیر مجاز محافظت میشود.
    -

بلاک کنترل فرآیند (PCB)

- اطلاعات فرآیندها و ریسمانها در سیستم عامل کجا ذخیره می‌شود؟
- سیستم عامل چگونه می‌داند که کدام ریسمان باید اجرا شود؟ لیست ریسمانها و فرآیندها در کجاست؟

- اطلاعات هر فرآیند در هسته سیستم عامل در یک ساختار (struct) با نام بلاک کنترل فرآیند (Process Control Block - PCB) ذخیره می‌شود
  - وضعیت فرآیند
  - اطلاعات کاربری که فرآیند را شروع کرده،آدرس فایل اجرایی، تقدم و ...
  - مدت زمان اجرا، مدت زمانی که از پردازنده استفاده کرده است، مدت زمانی که از پردازنده در حالت هسته استفاده کرده است و ...
  - میزان حافظه اشغال شده (حافظه مشترک، خصوصی و ...)

زمان‌بند (scheduler)

- هسته سیستم عامل دارای یک ریسمان زمان‌بند (scheduler) است
  - این بخش مشخص می‌کند در زمان تعویض زمینه ریسمان بعدی که باید اجرا شود کدام است
  - یک لیست از PCBها دارد
    - در حقیقت زمانی که یک فرآیند ایجاد می‌شود PCB مربوطه به لیست PCBهای زمان‌بند اضافه می‌شود

```
while(! powerOff) {

   if ( readyProcesses(PCBs) ) {
      nextPCB = selectNextProcess(PCBs);
      run( nextPCB );
   }
   else {
      run_idle_process();
   }
}
```

زمان‌بند (scheduler)

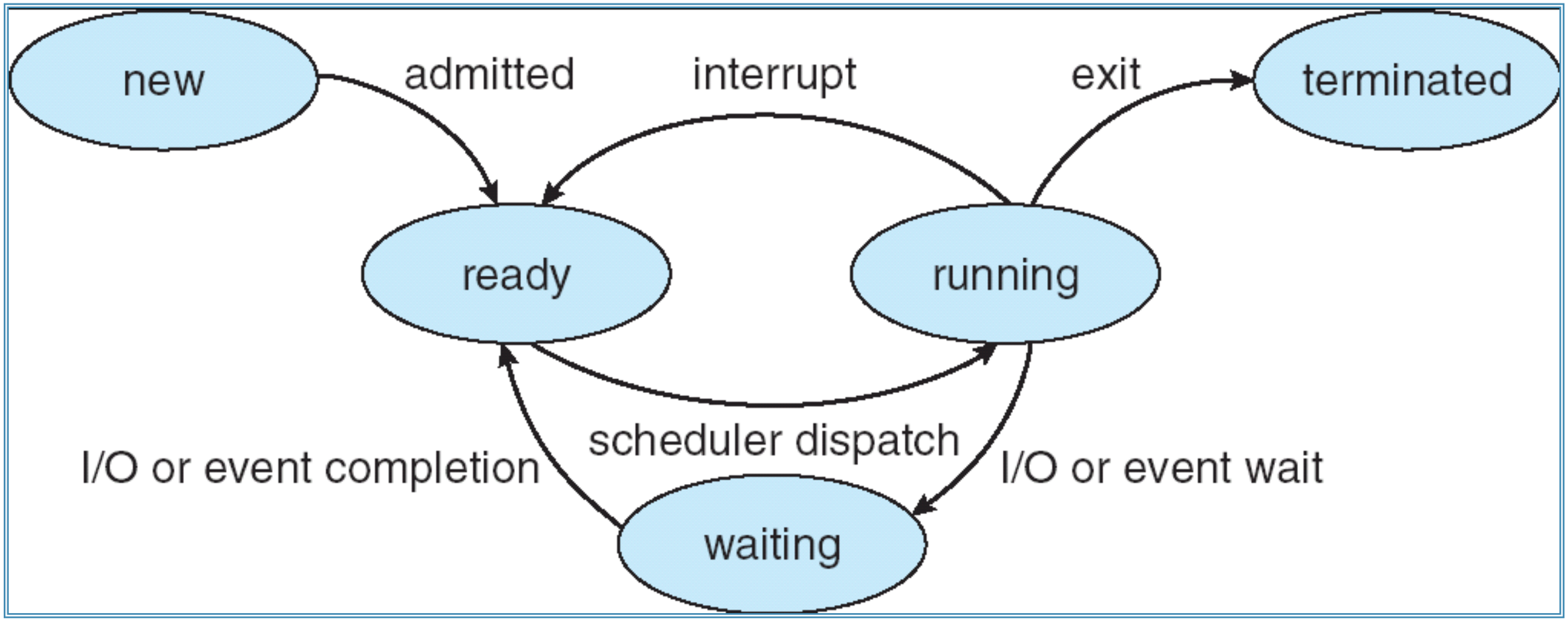
- تقریبا همه سیستم عاملها دارای یک فرآیند IDLE هستند

- زمانیکه در یکی از هسته‌ها چیزی برای اجرا نباشد این فرآیند اجرا میشود. این فرایند کمترین تقدم را داشته و همیشه در حالت ready است
  - بنابراین هیچگاه پردازنده بدون ریسمان قابل اجرا نخواهد بود.

- در این فرآیند معمولا مکانیزمهایی وجود دارد که پردازنده را به حالتهای مصرف کمتر سوق میدهد و برخی فعالیتها و قسمتهای مختلف پردازنده را خاموش میکند.

دوره زندگی فرآیندها

- فرآیندها در طول دوره حیاتشان در حالتهای مختلفی قرار می‌گیرند



دوره زندگی فرآیندها

- new – جدید: در ابتدا فرآیند که در حال ایجاد شدن است در این وضعیت قرار دارد.
  - آغاز کردن حافظه، مقدارهای اولیه برای ثباتها اختصاص پشته و heap و .... بعد که این فرآیند ایجاد شد و از ایجاد شدن صحیح آن مطمئن شدیم فرآیند به حالت ready میرود.

- ready – آماده اجرا: این فرآیند در صف فرآیندهای آماده قرار میگیرد. هر گاه که زمان‌بند این فرآیند را برای اجرا انتخاب کرد وضعیت ذخیره شده آن در پردازنده لود شده و اجرا میشود.

- running – در حال اجرا: فرآیند در حال اجرا روی پردازنده است. در این وضعیت ممکن است یک وقفه بیاید که باعث میشود اجرای این فرآیند متوقف شده و به صف آماده اجراها بازگردد

دوره زندگی فرآیندها

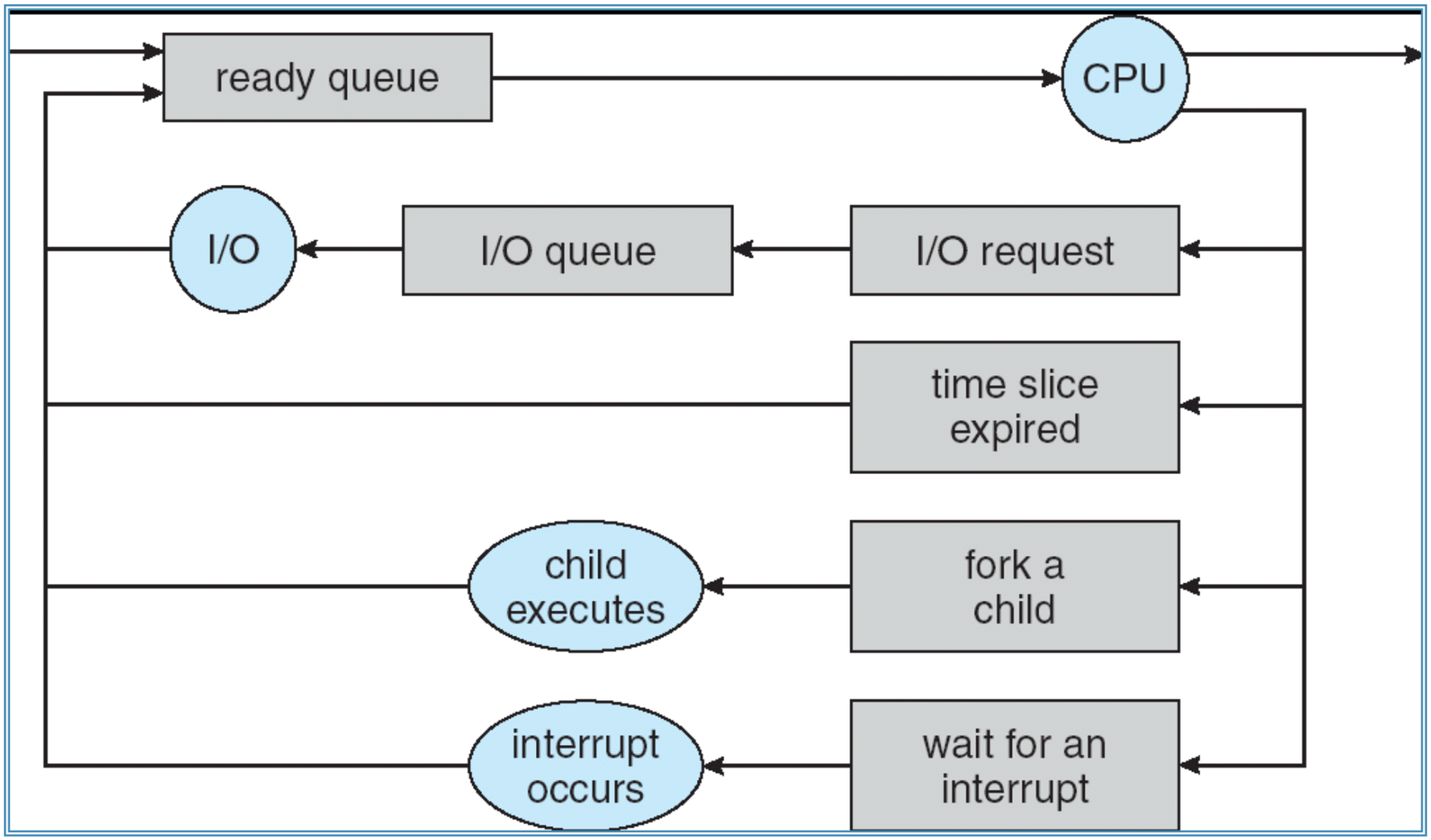
- waiting – بلاک شده یا منتظر: در این حالت فرآیند منتظر یک وقفه است (مثلا دیسک). زمانی که مقدار آماده شد فرآیند به صف آماده‌ها برمیگردد.
  - این حالت با ready متفاوت است چرا که اگر به فرآیندی که در این حالت است پردازنده بدهیم کاری برای اجرا ندارد

- terminated – تمام شده: زمانی که فرآیند تمام میشود فرآند به این حالت میرود. برای مدتی در این وضعیت هست چون ممکن است از مقدار برگشتی آن مثلا در فرآیند parent استفاده شود. برای همین مدتی در این وضعیت باقی میماند.

- تفاوت وضعیتهای waiting و ready چیست؟

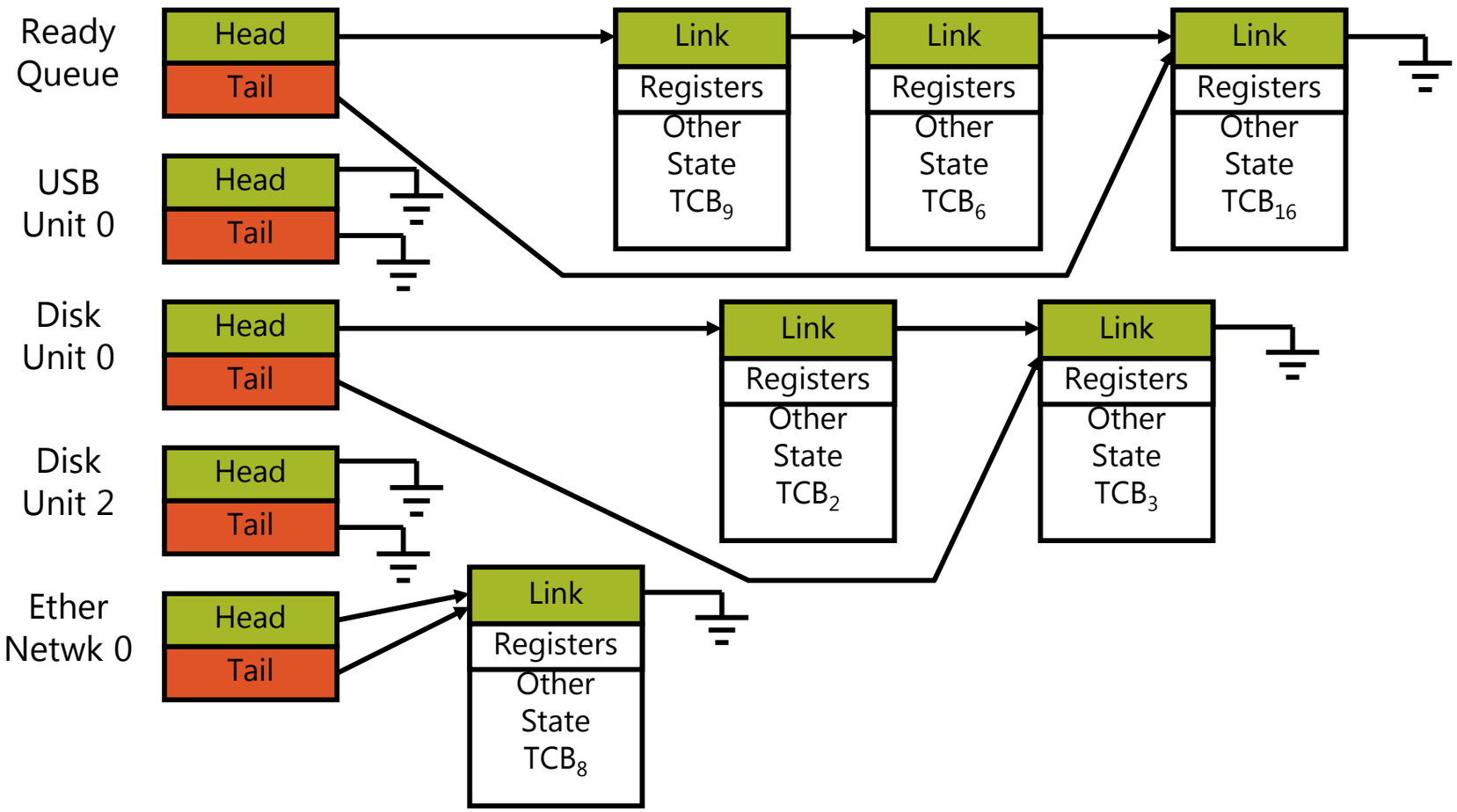
دوره زندگی فرآیندها

- PCB فرآیندها در طول مدت اجرا در صفهای مختلفی قرار می‌گیرد
- در حقیقت هر وضعیت برای خود یک صف مجزا دارد
- تغییر وضعیت یک فرآیند شامل جدا شدن PCB آن و الحاق به یک صف دیگر است
- ترتیب اینکه کدام فرآیندها از چه صفی به چه صفی جدا و ملحق شوند همه با زمان‌بند است



دوره زندگی فرآیندها

- برای منابع مختلف صفهای مختلفی در سیستم عامل تشکیل می‌شود تا زمان جستجوی فرآیندها به حداقل کاهش یابد
- دقت کنید که منظور از صف دقیقا روش FIFO نیست.



دسته‌بندی فرآیندها

- فرآیند می‌تواند IO Bound و یا CPU Bound باشد

- IO Bound : زمانی که پردازنده در اختیار فرآیند قرار می‌گیرد پردازش کمی انجام می‌دهد و بعد منتظر IO می‌ماند یعنی بیشتر مواقع از زمان اختصاص یافته استفاده نمی‌کند

- cpu bound: درصد عمده زمان اختصاص یافته برای استفاده از پردازنده را مصرف می‌کند و دائما در حال پردازش بوده و کمتر با تجهیزات جانبی کار دارد

دسته‌بندی فرآیندها

- فرآیند می‌تواند پیش‌زمینه (foreground) و یا پس‌زمینه (background) باشد
- فرآیندهای پیش‌زمینه عموما فرآیندهایی هستند که در تعامل با کاربر هستند و نیازمند زمان پاسخ‌های سریعتری هستند
- فرآیندهای پس‌زمینه فرآیندهایی هستند که نیاز به تعامل با کاربر ندارند و عموما در زمان‌بندی با تقدم پایین‌تر اجرا می‌شوند

دسته‌بندی فرآیندها

- چندبرنامگی (multiprogramming)
  - تعداد برنامه‌های آماده اجرا
  - این خاصیت برای افزایش بهره‌وری پردازنده لازم است تا زمانی که یک فرآیند به حالت انتظار رفت فرآیند دیگری آماده اجرا باشد
  - در حال حاضر فرض بر این است که تمام برنامه‌های آماده اجرا در حافظه هستند. بعدا توضیح می‌دهیم که لزوما اینطور نیست
- درجه چندبرنامگی
  - تعداد فرآیندهای آماده اجرا
- چندوظیفگی (multitasking)
  - اجرای فرآیندها با مکانیزم همزمانی (concurrency یا time sharing)
  - برای سیستمهایی که با کاربر تعامل (interactive systems) دارند این خاصیت لازم است.

مدیریت فرآیندها

مدیریت فرآیندها

- اطلاعات فرآیندها در سیستم عامل و در ساختاری با نام PCB ذخیره می‌شود
- وظیفه اصلی و بدیهی سیستم عامل اجرای آنهاست
- تمامی سیستم عامل‌ها رابطهای برنامه‌نویسی معینی برای دریافت و یا تغییر اطلاعات فرآیندها دارند که عمدتا این اطلاعات در PCB ذخیره شده است.

شناسه فرآیندها

```c
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#define BUFSIZE 1024
int main(int argc, char *argv[]){
   int c;

   pid_t pid = getpid();    /* get current processes PID */
   printf("My pid: %d\n", pid);

   c = fgetc(stdin);
   exit(0);
}
```

شناسه فرآیندها

- برنامه نوشته شده در زمان لود به فرآیند تبدیل می‌شود
- این برنامه شناسه فرآیند (PID) ایجاد شده را می‌خواند
- شناسه فرآیند یا PID عددی است که در سطح سیستم عامل برای هر فرآیند یکتاست و برای اشاره کردن به یک فرآیند در سطح سیستم عامل استفاده میشود.
  - pid_t: همان integer است.

- تابع exit را میتوان با return 0 هم تعویض کرد اما این کار nice تری است!

لیست فرآیندها

- برای دیدن لیست فرآیندها می‌توان از برنامه‌های گرافیکی در سیستم عاملها استفاده نمود
  - در ویندوز TaskManager
  - در مک Activity Monitor
- برنامه‌های غیرگرافیکی هم وجود دارد که زمانی که سیستم عاملها گرافیکی نیستند می‌توان از آنها استفاده نمود
- برای دیدن شماره فرآیندها در shell میتوان از دستور ps -ae استفاده کرد
- برنامه‌های htop, top محیطی شبیه به محیط گرافیکی دارند

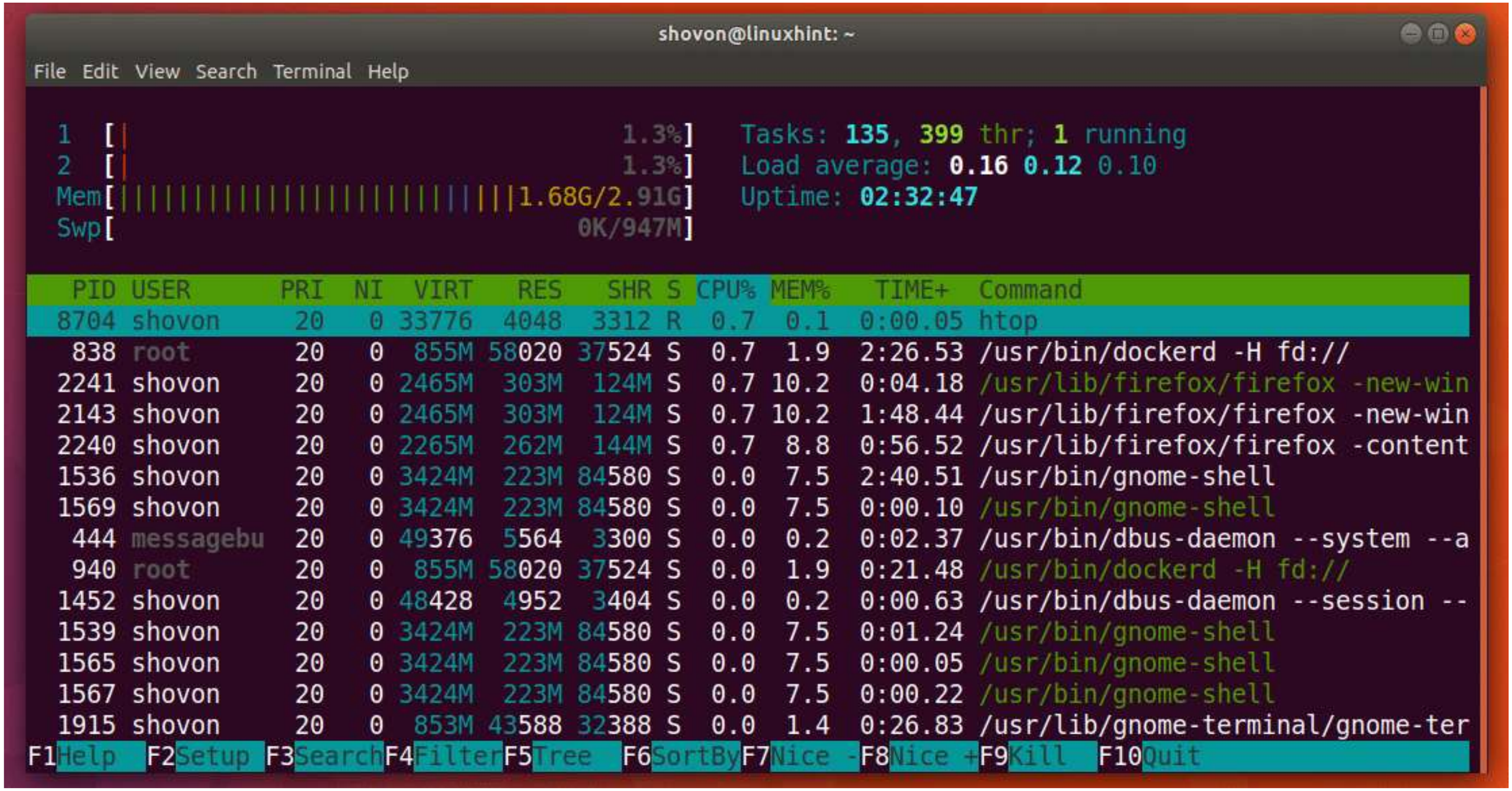
شناسه فرآیندها

نمایی از برنامه TOP

hamed — top — 124×48

```
Processes: 403 total, 2 running, 401 sleeping, 1822 threads                                 13:05:12
Load Avg: 2.40, 2.34, 2.41  CPU usage: 9.76% user, 7.38% sys, 82.85% idle
SharedLibs: 230M resident, 52M data, 16M linkedit. MemRegions: 117308 total, 2576M resident, 110M private, 1115M shared.
PhysMem: 8039M used (1570M wired), 152M unused.
VM: 1982G vsize, 1882M framework vsize, 20343626(0) swapins, 21068569(0) swapouts.
Networks: packets: 24843949/9120M in, 3477116/3177M out. Disks: 14459110/213G read, 6121386/198G written.

PID    COMMAND      %CPU TIME     #TH   #WQ  #PORTS MEM    PURG   CMPRS PGRP  PPID  STATE    BOOSTS          %CPU_ME
305    WindowServer 19.1 04:54:13 10    5    3097   380M-  4392K- 95M   305   1     sleeping *0[1]           0.00000
398    firefox      14.4 04:11:31 77    3    23733  878M   1616K  457M  398   1     sleeping *0[22853]       0.00000
0      kernel_task  8.5  08:51:49 168/4 0    0      56M    0B     0B    0     0     running   0[0]           0.00000
28207  top          4.9  00:01.51 1/1   0    27     4796K  0B     0B    28207 28194 running  *0[1]           0.00000
28208  screencaptur 4.4  00:00.54 2     1    54-    3488K- 620K   0B    487   487   sleeping *0[1]           0.10809
212    hidd         4.0  42:45.40 6     3    276    3060K  0B     868K  212   1     sleeping *0[1]           0.06699
11587  dotnet       3.4  03:04:04 25    0    62+    83M+   0B     41M   11583 11586 sleeping *0[1]           0.00000
27953  OneDrive     2.6  00:25.43 17    1    391    107M   116K   20M   27953 1     sleeping *0[227]         0.00000
187    ss_conn_serv 1.1  78:42.04 6     0    41     632K   0B     504K  187   1     sleeping *0[1]           0.00000
547    plugin-conta 1.1  01:43:22 42    1    8725   453M   0B     217M  398   398   sleeping *0[1802]        0.00000
28192  Terminal     1.0  00:01.71 7     2    240    24M    3392K  0B    28192 1     sleeping *0[31]          0.00074
25873  Microsoft Wo 0.8  08:17.08 16    8    1536   317M   3280K  166M  25873 1     sleeping *0[3699]        0.00000
28209  screencaptur 0.5  00:00.16 4     2    156-   4216K- 0B     0B    28209 1     sleeping *0[244+]        0.07649
397    AXVisualSupp 0.5  14:01.39 5     1    196    7760K  0B     4048K 397   1     sleeping *0[1]           0.00000
3331   plugin-conta 0.2  32:09.81 35    1    1483   152M   0B     101M  398   398   sleeping *0[1511]        0.00000
22312  plugin-conta 0.1  11:25.83 30    1    642    254M   0B     99M   398   398   sleeping *0[650]         0.00000
166    powerd       0.1  03:37.59 3     2    113    2208K  0B     660K  166   1     sleeping *0[1]           0.00000
189    launchservic 0.1  01:59.08 4     3    581-   10M-   0B     5456K 189   1     sleeping *0[262227+]     0.00000
2896   plugin-conta 0.0  19:03.22 31    1    1297   156M   0B     119M  398   398   sleeping *0[1564]        0.00000
27414  Microsoft Ex 0.0  07:30.24 21    8    890+   142M+  256K   124M- 27414 1     sleeping  0[2090]        0.00000
27951  mdworker_sha 0.0  00:00.73 4     1    61     6396K  0B     1276K 27951 1     sleeping *0[1]           0.11341
1544   plugin-conta 0.0  17:35.65 33    1    1941   189M   0B     101M  398   398   sleeping *0[1697]        0.00000
1542   plugin-conta 0.0  29:42.48 33    1    1736-  239M-  0B     155M  398   398   sleeping *0[1695]        0.00000
16184  Finder       0.0  05:54.22 14    3    1092   172M   156K   79M   16184 1     sleeping *0[21419]       0.00000
650    storagekitd  0.0  03:36.35 8     4    80     4848K  0B     4028K 650   1     sleeping  0[2483]        0.00000
170    logd         0.0  06:11.10 4     3    1229   33M    0B     21M   170   1     sleeping *0[1]           0.00000
157    fseventsd    0.0  04:17.40 11    1    259    6952K  0B     4396K 157   1     sleeping *0[1]           0.00000
487    SystemUIServ 0.0  12:31.75 3     1    309-   24M-   0B     11M   487   1     sleeping *0[20581]       0.00000
433    lockoutagent 0.0  02:23.57 4     3    73     3832K  0B     2880K 433   1     sleeping *0[1]           0.00000
215    AirPlayXPCHe 0.0  01:50.84 7     4    133    3924K  0B     2724K 215   1     sleeping *0[1]           0.00000
6052   SafariBookma 0.0  01:40.64 6     4    74     8912K  0B     7600K 6052  1     sleeping *0[1]           0.00000
186    opendirector 0.0  02:54.50 6     5    1093-  8880K- 60K    3476K 186   1     sleeping *0[1]           0.00000
5452   com.apple.ge 0.0  02:10.16 4     2    112    10M+   128K   5940K 5452  1     sleeping  0[15054]       0.00000
1      launchd      0.0  20:18.84 4     3    3692-  25M-   0B     10M   1     0     sleeping  0[0]           0.00000
341    runningboard 0.0  00:28.67 4     3    170-   3440K- 0B     876K  341   1     sleeping *3[1]           0.00000
28172  FinderSync   0.0  00:00.28 5     1    173    8148K  4096B  384K  28172 1     sleeping *0[42]          0.00000
362    mds_stores   0.0  32:42.21 4     2    108-   91M-   328K   76M   362   1     sleeping *0[1]           0.00000
358    locationd    0.0  06:20.77 4     2    232    8204K  128K   4028K 358   1     sleeping  0[45855]       0.00000
224    loginwindow  0.0  01:39.67 4     1    431-   32M-   0B     9736K 224   1     sleeping *0[10804]       0.00000
497    AMPDeviceDis 0.0  00:00.79 4     2    131-   2200K- 0B     1444K 497   1     sleeping  0[1246]        0.00000
```

شناسه فرآیندها



نمایی از برنامه htop نسخه پیشرفته‌تر از top

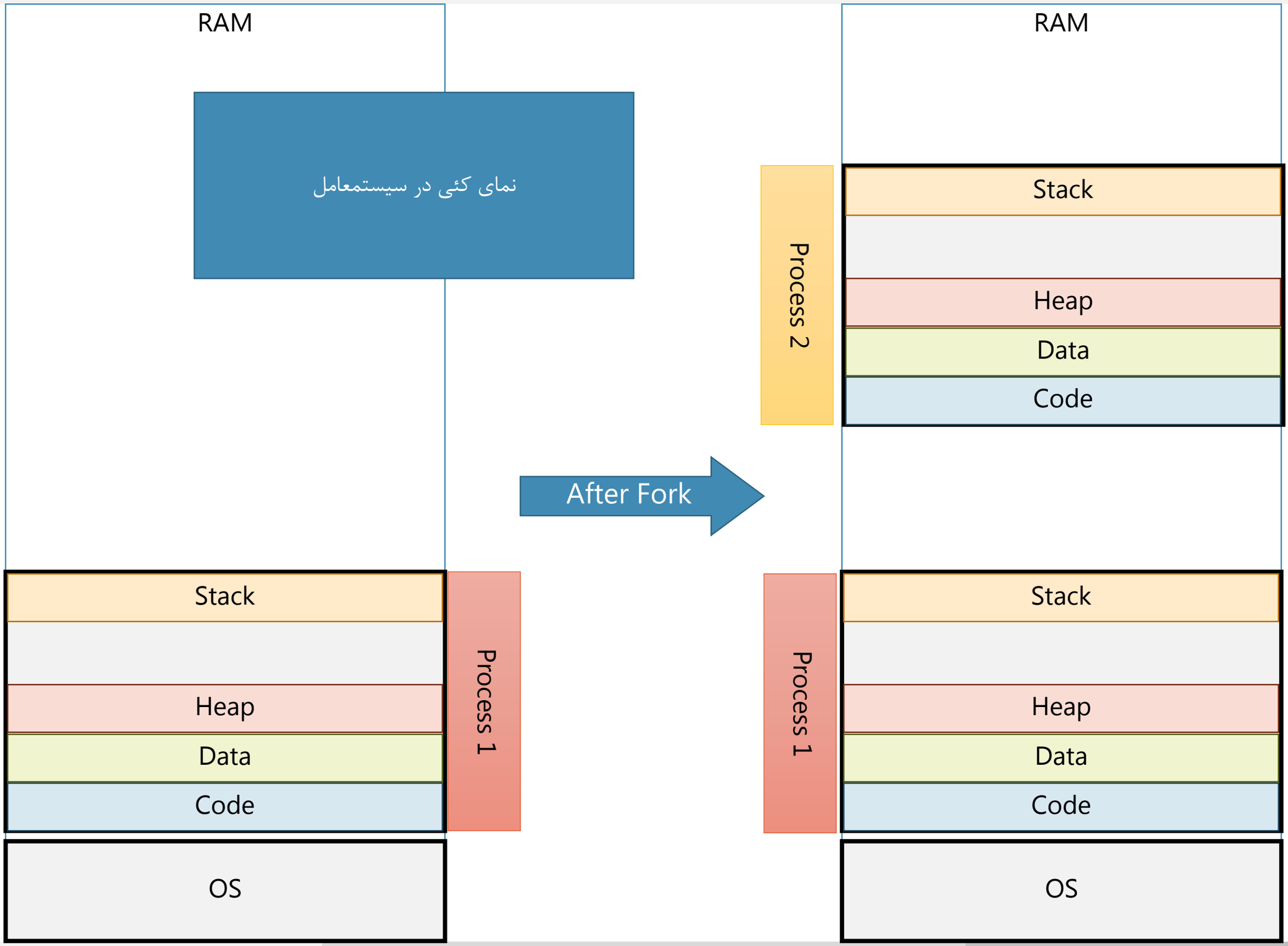
ایجاد فرآیند

- ایجاد فرآیند در لینوکس (یا سیستم عاملهای مبتنی بر UNIX) با دستور ()fork انجام می‌شود

- این دستور یک کپی کامل از فرآیند فعلی ایجاد می‌کند
  - تمام محتوای حافظه (کد، داده،پشته، heap و ...)
    - بعدا توضیح می‌دهیم که این تابع چطور بهینه پیاده‌سازی می‌شود
  - هر چیزی که در PCB فرآیند فعلی وجود دارد
    - مقدار ثبات PC
    - مقادیر ثباتهای Base, Bound, Stack Pointer (SP) طبیعتا برای فرآیند جدید بروزرسانی می‌شود
    - فرآیند جدید PID جدید دارد

- فرآیند اول فرآیند والد (Parent Process) و فرآیند دوم فرآیند فرزند (Child Process) نامیده می‌شود
- چون مقدار ثبات PC کپی می‌شود هر دو فرآیند دقیقا از همانجایی که fork انجام شده اجرا می‌شوند.

ایجاد فرآیند



ایجاد فرآیند

Process 1

Process 2

```c
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
int main(int argc, char *argv[]){

  fork();
  // from this point two
  // processes in separate space
  // execute the following code

  printf("Hello \n");
}
```

```
>./code
Hello
Hello
```

ایجاد فرآیند

Process 1

```c
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
int main(int argc, char *argv[]){

  fork();
  // from this point two
  // processes in separate space
  // execute the following code

  printf("Hello \n");
}
```

Process 2

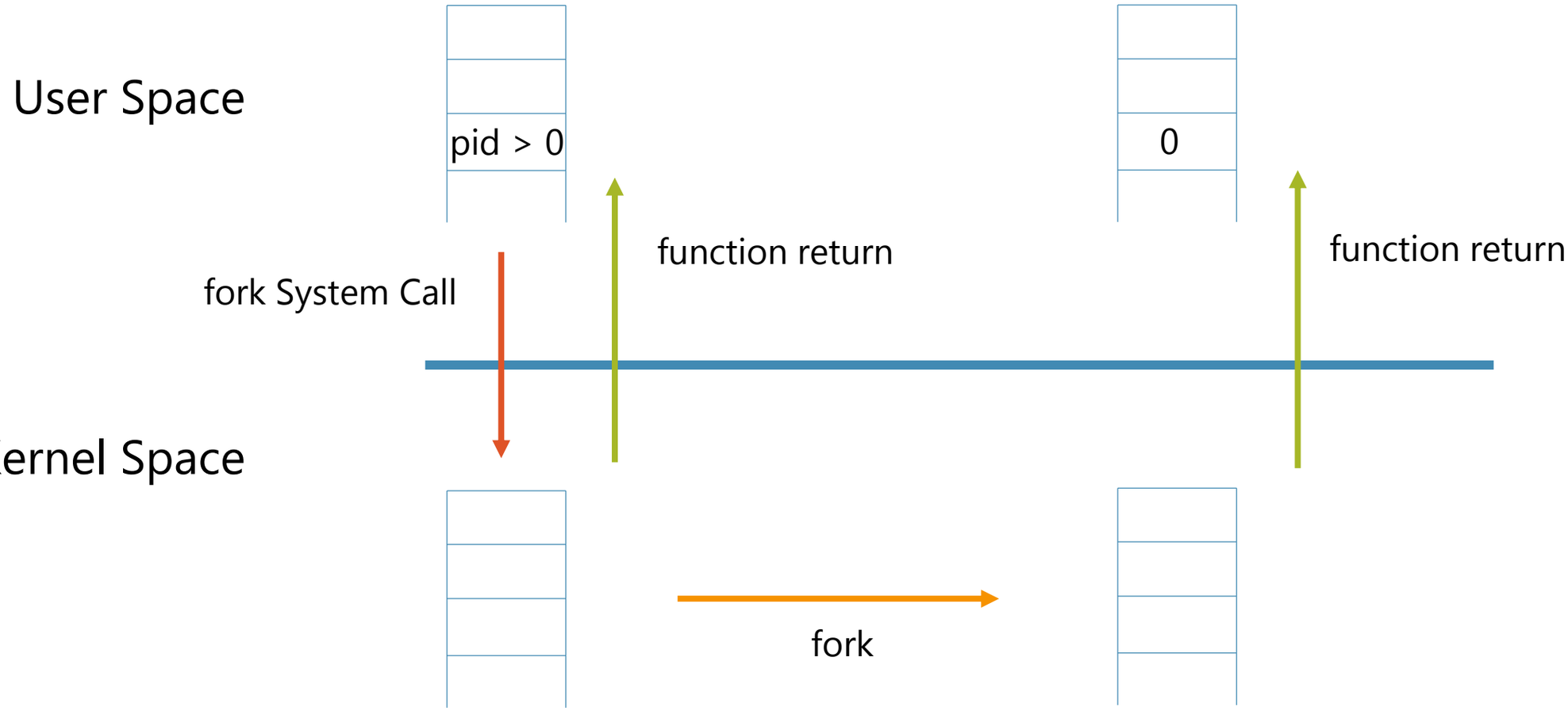
```c
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
int main(int argc, char *argv[]){

  fork();
  // from this point two
  // processes in separate space
  // execute the following code

  printf("Hello \n");
}
```

```
>./code
Hello
Hello
```

ایجاد فرآیند

- تابع fork تابع جالبی است. زمانیکه فراخوانی میشود دوبار باز میگردد (return).
  - تابع fork تابع سیستمی است یعنی در سیستم عامل انجام می‌شود بنابراین از پشته هسته برای کار استفاده می‌شود.
  - در زمان اجرا همه چیز از فرآیند اول کپی می‌شود حتی پشته هسته
  - زمانی کار fork تمام شد بدیهی است که باید نتیجه در پشته کاربری نوشته شده و اجرا به حالت کاربری بازگردد
  - اما در این نقطه یک فرآیند جدید هم ساخته شده که هر دو مقادیر را در پشته‌های کاربری خاص خود کپی کرده و به حالت اجرای کاربری return می‌کنند.
  - نکته جالب دیگر اینکه مقادیر برگشتی در دو فرآیند با هم متفاوت است

ایجاد فرآیند



ایجاد فرآیند

- اگر بخواهیم که در فرایند جدید کار دیگری انجام دهیم و در فرآیند والد کار دیگر باید به مقدار برگشتی از تابع fork توجه کنیم.
  - اگر چه هر دو تابع fork در فرآیند والد و فرزند به حالت اجرایی کاربری برگشت می‌کنند ولی نتیجه تابع در دو فرآیند متفاوت است

- اگر مقدار برگشتی بزرگتر از 0 بود این بازگشت در فرآیند والد اتفاق افتاده است و مقدار برگشتی تابع PID فرآیند فرزند است
- اگر مقدار برگشتی مساوی 0 بود این کار بازگشت در فرآیند والد اتفاق افتاده است
- اگر مقدار برگشتی کوچکتر از 0 بود یعنی تابع fork در اجرا دچار خطا شده است

ایجاد فرآیند

```c
int main(int argc, char *argv[]){

  pid_t pid = fork();

  if(pid > 0){
    printf("Hello from Parent\n");
    printf("Child process Id=%d\n", pid);
  }
  else if(pid == 0){
    printf("Hello from Child\n");
    printf("My Id=%d\n", getpid());
  }
  else { //pid<0
      printf("Fork Error!\n");
  }
}
```

ایجاد فرآیند

Parent Process

```
7  int main(int argc, char *argv[]){
8
9    pid_t pid = fork();
10
11   if(pid > 0){
12     printf("Hello from Parent\n");
13     printf("Child process Id=%d\n", pid);
14   }
15   else if(pid == 0){
16     printf("Hello from Child\n");
17     printf("My Id=%d\n", getpid());
18   }
19   else { //pid<0
20       printf("Fork Error!\n");
21   }
22 }
```

Child Process

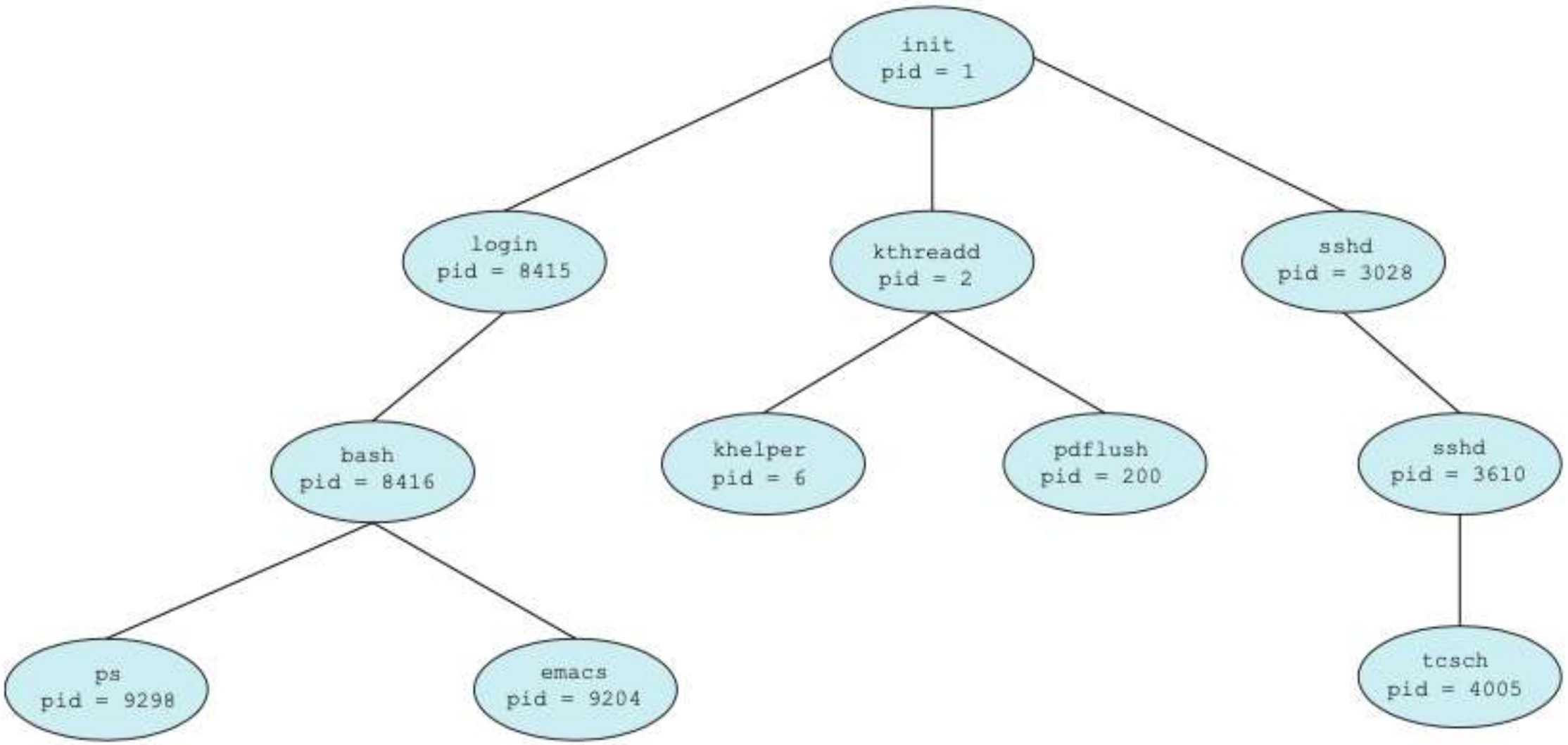
```
7  int main(int argc, char *argv[]){
8
9    pid_t pid = fork();
10
11   if(pid > 0){
12     printf("Hello from Parent\n");
13     printf("Child process Id=%d\n", pid);
14   }
15   else if(pid == 0){
16     printf("Hello from Child\n");
17     printf("My Id=%d\n", getpid());
18   }
19   else { //pid<0
20       printf("Fork Error!\n");
21   }
22 }
```

ایجاد فرآیند

- در کد قبل کدام رشته زودتر چاپ می‌شود
- به عبارت دیگر فرآیند والد زودتر اجرا می‌شود یا فرآیند فرزند؟

- با توجه به عملکرد زمان‌بند اصلا مشخص نیست چون پس از fork هر دو فرآیند آماده به کار (ready) هستند و در صف قرار می‌گیرند و اینکه کدام زودتر انجام شود مشخص نیست.

ایجاد فرآیند

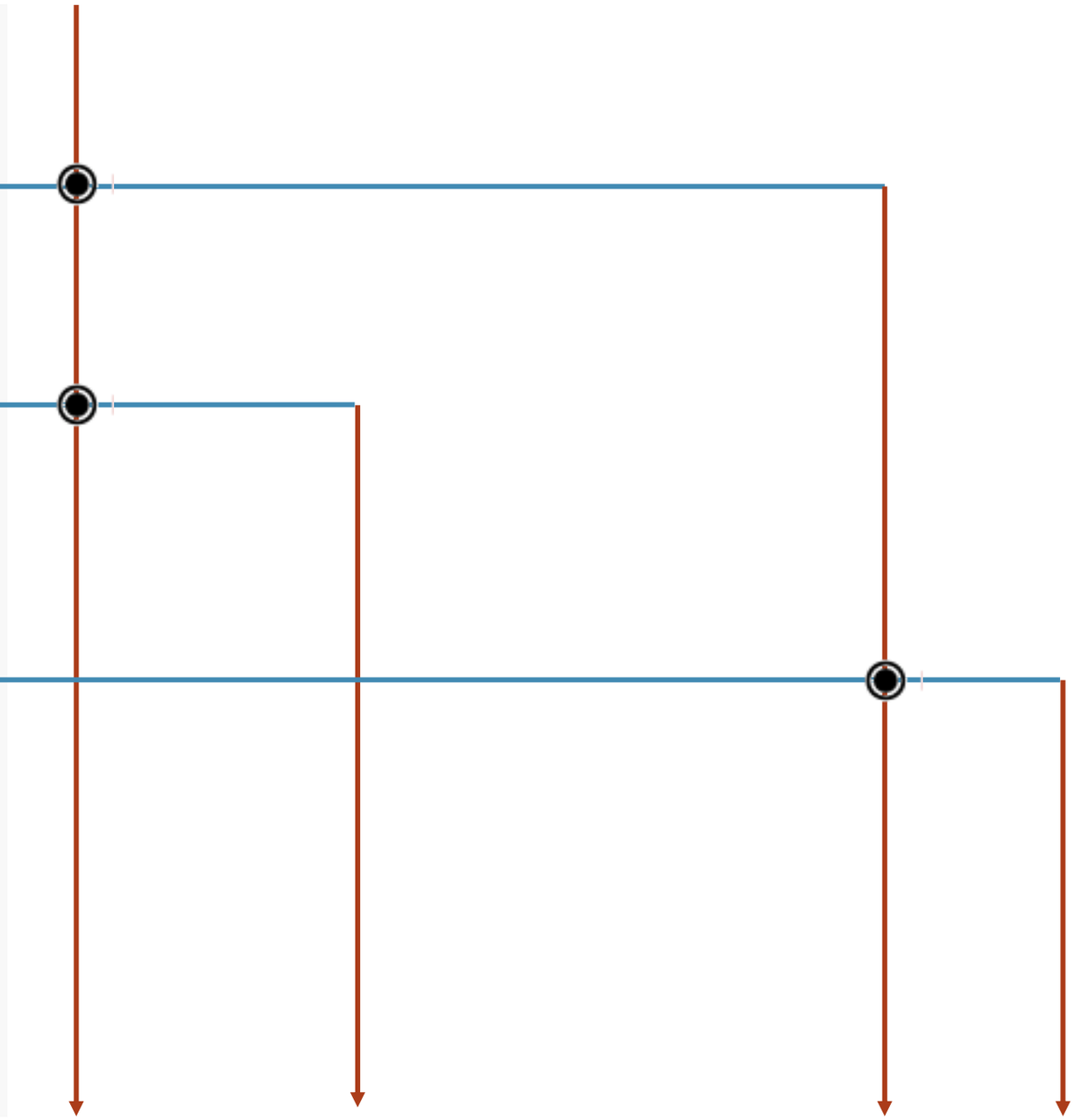


- در لینوکس هم فرآیندها از یک والد به نام init منشعب می‌شوند. init اولین فرآیند است که توسط سیستم عامل ساخته می‌شود.
- سایر فرآیندها همه بوسیله fork از این فرآیند ساخته می‌شوند

ایجاد فرآیند

- ◉: محل انشعاب فرآیندها

```
7  int main(int argc, char *argv[]){
8
9    pid_t pid = fork();
10
11   if(pid > 0){
12     fork();
13   }
14   else if(pid == 0){
15     printf("Hello from Child\n");
16     fork();
17   }
18   else { //pid<0
19       printf("Fork Error!\n");
20   }
21 }
```



ایجاد فرآیندها

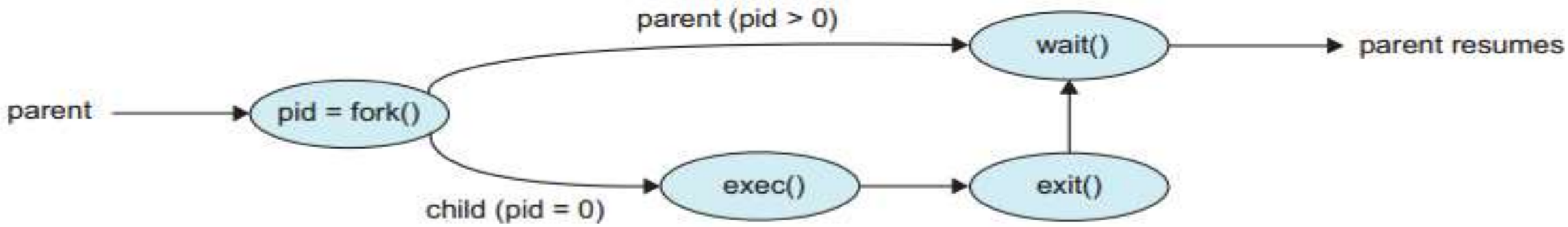
- خروجی این برنامه چیست؟
- آیا ممکن است این برنامه هیچگاه تمام نشود؟

```c
int i=50;
cpid = fork();
if (cpid > 0) {
    mypid = getpid();
    printf("[%d] parent of [%d]\n", mypid, cpid);
    for (; i<100; i++) {
      printf("[%d] parent: %d\n", mypid, i);
      sleep(1);
    }
}
else if (cpid == 0) {
    mypid = getpid();
    printf("[%d] child\n", mypid);
    for (; i>-100; i--) {
      printf("[%d] child: %d\n", mypid, i);
      sleep(1);
    }
}
```

مدیریت فرآیندها

- تابع wait در فرآیند والد فراخوانی می‌شود که باعث می‌شود والد بلاک شده و منتظر اتمام فرآیند فرزند شود

- تابع exec کل فضای آدرس را با یک برنامه دیگر پر می‌کند. یعنی یک برنامه دیگر روی آنچه از والد کپی شده جایگزین می‌شود




```c
int main() {
  pid t pid;

  pid = fork();
  if (pid < 0) {
    fprintf(stderr, "Fork Failed");
    return 1;
  }
  else if (pid == 0) { /* child process  */
    execlp("/bin/ls","ls",NULL);
  }
  else { /* parent process  */
    /* wait for the child to complete  */
    wait(NULL);
    printf("Child Complete");
  }
  return 0;
}
```

مدیریت فرآیندها

- تابع execlp از خانواده توابع exec است که بیشتر در نوع و تعداد پارامترها با هم فرق دارند.

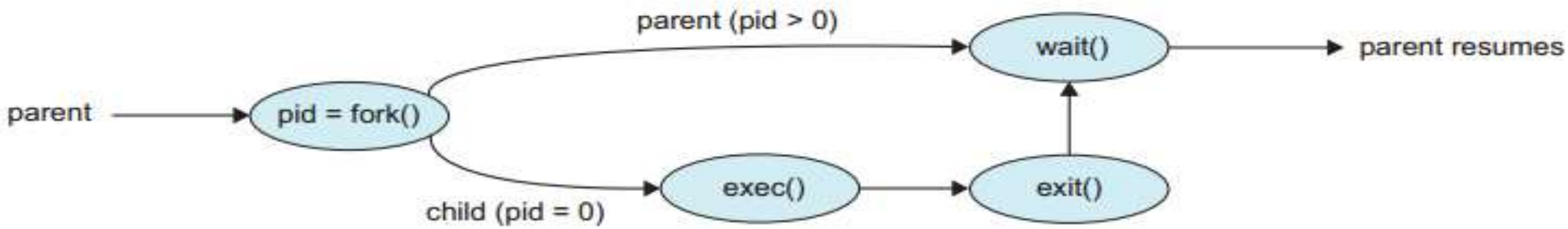
- فرآیند **زامبی** (Zombie)
- فرآیند فرزندی که تمام شده ولی فرآیند والد هنوز wait را صدا نکرده است.
- این حالت می‌تواند موقتی باشد و فرآیند والد بدلیل اجرای یک کار طولانی هنوز به دستور wait نرسیده باشد

- فرآیند یتیم (Orphan):
- فرآیند فرزندی که والدش تمام شده و wait را صدا نکرده است.
- در این حالت معمولا این فرآیندها به فرآیند init نسبت داده شده و اگر تمام شوند از طریق والد جدید آزاد می‌شوند.

برنامه shell

- همه سیستم عاملها دارای یک برنامه ارتباط با کاربر به شکل غیر گرافیکی هستند که در خانواده Unix با نام shell و در ویندوز با نام خط فرمان (command line) شناخته می‌شوند.

- shell یک برنامه کنترل و مدیریت کارهاست که این کارها توسط کاربر ایجاد و مدیریت می‌شوند
  - برنامه قبل در حقیقت یک نسخه ساده شده از پیاده‌سازی shell است.
  - زمانی که کاربر یک دستور خط فرمان وارد می‌کند یک فرآیند جدید fork شده و بعد دستور وارد شده exec میشود.



Activity Monitor (All Processes, Hierarchically)

CPU | Memory | Energy | Disk | Network | Search

Process Name	% CPU	CPU Time	Threads	Idle Wake Ups	% GPU	GPU Time	PID	User
suhelperd	0.0	0.05	2	0	0.0	0.00	633	root
swcd	0.0	0.52	2	0	0.0	0.00	695	hamed
symptomsd	0.0	5.37	2	0	0.0	0.00	308	_networkd
sysdiagnose	0.0	0.04	2	0	0.0	0.00	1667	root
syslogd	0.0	1.04	3	0	0.0	0.00	151	root
sysmond	0.6	2.30	3	1	0.0	0.00	490	root
syspolicyd	0.0	4.43	3	0	0.0	0.00	221	root
system_installd	0.0	0.03	2	0	0.0	0.00	607	root
systemsoundserverd	0.0	0.16	4	0	0.0	0.00	426	root
systemstats	0.0	6.92	4	0	0.0	0.00	163	root
systemstats	0.0	0.10	4	0	0.0	0.00	641	root
SystemUIServer	0.0	7.95	3	0	0.0	0.00	422	hamed
talagent	0.0	0.89	2	0	0.0	0.00	420	hamed
taskgated	0.0	0.05	1	0	0.0	0.00	217	root
tccd	0.0	6.17	2	0	0.0	0.00	220	root
tccd	0.0	11.01	2	0	0.0	0.00	415	hamed
Terminal	1.1	34.02	6	4	0.0	0.00	412	hamed
login	0.0	0.05	2	0	0.0	0.00	1734	root
bash	0.0	0.03	1	0	0.0	0.00	1735	hamed
top	5.0	4.47	1	1	0.0	0.00	1745	root
login	0.0	0.03	2	0	0.0	0.00	507	root
bash	0.0	0.03	1	0	0.0	0.00	511	hamed
sudo	0.0	0.11	2	0	0.0	0.00	584	root
bash	0.0	0.01	1	0	0.0	0.00	592	root
dotnet	0.0	14.52	27	0	0.0	0.00	594	root

تجربه عملی

- نمایش عملکرد top و ps-ae
- نمایش آمار اجرای فرآیندها

- `ps -aefc -o utime,pcpu,state`
  - `-ae : all processes`
  - `f: show time information`
  - `c: show name of processes not the full path`
  - `-o: additional info`
  - `utime: user space time`
  - `pcpu: CPU percent usage`
  - `state: process state`

- در ویندوز می‌توان از برنامه Process Explorer استفاده کرد.

~/Downloads/DnsServerPortable — -bash ~/Downloads/DnsServerPortable — dotnet

```
FreshAir:DnsServerPortable hamed$ ps -ae
  PID TTY           TIME CMD
    1 ??         0:51.98 /sbin/launchd
  151 ??         0:01.05 /usr/sbin/syslogd
  152 ??         0:06.31 /usr/libexec/UserEventAgent (System)
  155 ??         0:00.54 /System/Library/PrivateFrameworks/Uninstall.framework/Resources/uninstalld
  156 ??         0:08.38 /usr/libexec/kextd
  157 ??         0:25.67 /System/Library/Frameworks/CoreServices.framework/Versions/A/Frameworks/FSEvents.frame
  159 ??         0:00.26 /System/Library/PrivateFrameworks/MediaRemote.framework/Support/mediaremoted
  163 ??         0:06.93 /usr/sbin/systemstats --daemon
  164 ??         0:04.80 /usr/libexec/configd
  165 ??         0:00.04 endpointsecurityd
  166 ??         0:07.21 /System/Library/CoreServices/powerd.bundle/powerd
  170 ??         0:21.87 /usr/libexec/logd
  171 ??         0:00.13 /usr/libexec/keybagd -t 15
  174 ??         0:00.58 /usr/libexec/watchdogd
  178 ??         0:18.51 /System/Library/Frameworks/CoreServices.framework/Frameworks/Metadata.framework/Suppor
  179 ??         0:00.08 /System/Library/CoreServices/iconservicesd
  180 ??         0:02.11 /usr/libexec/diskarbitrationd
  183 ??         0:01.50 /usr/libexec/coreduetd
  186 ??         0:09.72 /usr/libexec/opendirectoryd
  187 ??         2:30.43 /Library/LaunchAgents/ss_conn_service
  188 ??         0:07.10 /System/Library/PrivateFrameworks/ApplePushService.framework/apsd
  189 ??         0:07.28 /System/Library/CoreServices/launchservicesd
  190 ??         0:01.09 /usr/libexec/timed
  191 ??         0:00.03 /Applications/OneDrive.app/Contents/StandaloneUpdaterDaemon.xpc/Contents/MacOS/Standal
  192 ??         0:00.11 /System/Library/PrivateFrameworks/MobileDevice.framework/Versions/A/Resources/usbmuxd
  193 ??         0:06.93 /usr/sbin/securityd -i
  194 ??         0:00.01 auditd -l
  200 ??         0:00.03 autofsd
  201 ??         0:00.14 /usr/libexec/displaypolicyd -k 1
  203 ??         0:07.78 /usr/libexec/dasd
  207 ??         0:00.07 /System/Library/CoreServices/logind
  208 ??         0:00.72 /System/Library/PrivateFrameworks/GenerationalStorage.framework/Versions/A/Support/rev
  209 ??         0:00.02 /usr/sbin/KernelEventAgent
  211 ??         0:01.69 /usr/sbin/bluetoothd
  212 ??         1:44.71 /usr/libexec/hidd
  214 ??         0:06.67 /usr/libexec/corebrightnessd --launchd
  215 ??         0:02.62 /usr/libexec/AirPlayXPCHelper
  216 ??         0:07.12 /usr/sbin/notifyd
  217 ??         0:00.05 /usr/libexec/taskgated
  218 ??         0:00.27 /usr/sbin/distnoted daemon
  219 ??         0:01.76 /usr/libexec/amfid
  220 ??         0:06.21 /System/Library/PrivateFrameworks/TCC.framework/Resources/tccd system
  221 ??         0:04.43 /usr/libexec/syspolicyd
  222 ??         0:00.11 aslmanager
  223 ??         0:02.81 /usr/sbin/cfprefsd daemon
  224 ??         0:01.03 /System/Library/CoreServices/coreservicesd
```

~/Downloads/DnsServerPortable — -bash

```
FreshAir:DnsServerPortable hamed$ ps -aefc -o utime,pcpu,state
  UID   PID  PPID   C STIME   TTY           TIME CMD                         UTIME  %CPU STAT
    0     1     0   0 Sun03PM ??         0:52.34 launchd                   0:07.57   0.0 Ss
    0   151     1   0 Sun03PM ??         0:01.06 syslogd                   0:00.36   0.0 Ss
    0   152     1   0 Sun03PM ??         0:06.33 UserEventAgent            0:02.49   0.0 Ss
    0   155     1   0 Sun03PM ??         0:00.54 uninstalld                0:00.16   0.0 Ss
    0   156     1   0 Sun03PM ??         0:08.38 kextd                     0:04.81   0.0 Ss
    0   157     1   0 Sun03PM ??         0:25.70 fseventsd                 0:09.06   0.0 Ss
    0   159     1   0 Sun03PM ??         0:00.26 mediaremoted              0:00.09   0.0 Ss
    0   163     1   0 Sun03PM ??         0:07.20 systemstats               0:03.24   0.0 Ss
    0   164     1   0 Sun03PM ??         0:04.81 configd                   0:02.03   0.0 Ss
    0   165     1   0 Sun03PM ??         0:00.04 endpointsecurity          0:00.03   0.0 Ss
    0   166     1   0 Sun03PM ??         0:07.35 powerd                    0:02.98   0.0 Ss
    0   170     1   0 Sun03PM ??         0:21.99 logd                      0:13.04   0.0 Ss
    0   171     1   0 Sun03PM ??         0:00.13 keybagd                   0:00.06   0.0 Ss
    0   174     1   0 Sun03PM ??         0:00.58 watchdogd                 0:00.27   0.0 Ss
    0   178     1   0 Sun03PM ??         0:18.57 mds                       0:06.97   0.0 Ss
  240   179     1   0 Sun03PM ??         0:00.08 iconservicesd             0:00.04   0.0 Ss
    0   180     1   0 Sun03PM ??         0:02.12 diskarbitrationd          0:01.32   0.0 Ss
    0   183     1   0 Sun03PM ??         0:01.59 coreduetd                 0:00.99   0.0 Ss
    0   186     1   0 Sun03PM ??         0:09.75 opendirectoryd            0:05.34   0.0 Ss
    0   187     1   0 Sun03PM ??         2:31.50 ss_conn_service           1:21.45   0.7 Ss
    0   188     1   0 Sun03PM ??         0:07.19 apsd                      0:04.69   0.0 Ss
    0   189     1   0 Sun03PM ??         0:08.05 launchservicesd           0:04.76   6.6 Ss
  266   190     1   0 Sun03PM ??         0:01.10 timed                     0:00.51   0.0 Ss
    0   191     1   0 Sun03PM ??         0:00.03 StandaloneUpdate          0:00.01   0.0 Ss
  213   192     1   0 Sun03PM ??         0:00.11 usbmuxd                   0:00.03   0.0 Ss
    0   193     1   0 Sun03PM ??         0:06.93 securityd                 0:05.19   0.0 Ss
    0   194     1   0 Sun03PM ??         0:00.01 auditd                    0:00.00   0.0 Ss
    0   200     1   0 Sun03PM ??         0:00.03 autofsd                   0:00.00   0.0 Ss
  244   201     1   0 Sun03PM ??         0:00.14 displaypolicyd            0:00.02   0.0 Ss
    0   203     1   0 Sun03PM ??         0:07.85 dasd                      0:05.48   0.0 Ss
    0   207     1   0 Sun03PM ??         0:00.07 logind                    0:00.02   0.0 Ss
    0   208     1   0 Sun03PM ??         0:00.72 revisiond                 0:00.25   0.0 Ss
    0   209     1   0 Sun03PM ??         0:00.02 KernelEventAgent          0:00.00   0.0 Ss
    0   211     1   0 Sun03PM ??         0:01.70 bluetoothd                0:00.58   0.0 Ss
  261   212     1   0 Sun03PM ??         1:47.47 hidd                      1:07.55   0.0 Ss
    0   214     1   0 Sun03PM ??         0:06.68 corebrightnessd           0:03.72   0.0 Ss
    0   215     1   0 Sun03PM ??         0:02.63 AirPlayXPCHelper          0:00.87   0.0 Ss
    0   216     1   0 Sun03PM ??         0:07.13 notifyd                   0:02.13   0.0 Ss
    0   217     1   0 Sun03PM ??         0:00.05 taskgated                 0:00.01   0.0 Ss
  241   218     1   0 Sun03PM ??         0:00.27 distnoted                 0:00.15   0.0 Ss
    0   219     1   0 Sun03PM ??         0:01.76 amfid                     0:01.47   0.0 Ss
    0   220     1   0 Sun03PM ??         0:06.27 tccd                      0:04.38   0.0 Ss
    0   221     1   0 Sun03PM ??         0:04.44 syspolicyd                0:03.43   0.0 Ss
    0   222     1   0 Sun03PM ??         0:00.11 aslmanager                0:00.03   0.0 Ss
    0   223     1   0 Sun03PM ??         0:02.82 cfprefsd                  0:01.30   0.0 Ss
    0   224     1   0 Sun03PM ??         0:01.05 coreservicesd             0:00.61   0.0 Ss
```

Process Explorer - Sysinternals: www.sysinternals.com [HAMEDFEFD\hamed]

File Options View Process Find Users Help

Process	PID	CPU	Description	Company Name	Page Faults	Virtual Size	Working Set	Peak Working Set	PF Delta	Min Working Set	Max Working Set
svchost.exe	400				17,086,006	539,632 K	78,260 K	498,736 K		K	K
wuauclt.exe	4416		Windows Update	Microsoft Corporation	2,759	76,364 K	1,660 K	7,508 K		200 K	1,380 K
svchost.exe	744				3,743	37,280 K	3,716 K	7,324 K		K	K
svchost.exe	1108				121,202	406,536 K	17,872 K	56,456 K		K	K
spoolsv.exe	1228				148,619	82,352 K	5,892 K	15,012 K		K	K
svchost.exe	1280				71,599	72,928 K	10,816 K	58,380 K		K	K
svchost.exe	1404						2,272 K	10,084 K		K	K
OfficeClickToRun.exe	1432						29,264 K	69,836 K		K	K
svchost.exe	1552						2,148 K	7,712 K		K	K
FoxitConnectedPDFService.exe	1592						6,036 K	14,496 K		K	K
SMSvcHost.exe	1688						69,092 K	69,120 K		K	K
NitroPDFReaderDriverService3x64.exe	1812						912 K	3,804 K		K	K
coherence.exe	1852						884 K	3,740 K		K	K
coherence.exe	1916						260 K	3,288 K		K	K
coherence.exe	1948						240 K	3,580 K		K	K
prl_tools_service.exe	1900						4,248 K	8,680 K		K	K
prl_tools.exe	1976						6,724 K	31,052 K		K	K
prl_cc.exe	3888		Parallels Contr				12,664 K	22,556 K		200 K	1,380 K
svchost.exe	1968						1,056 K	3,712 K		K	K
dllhost.exe	1652						1,632 K	6,940 K		K	K
ss_conn_service.exe	2076						1,240 K	4,964 K		K	K
ss_conn_service2.exe	2188						1,248 K	5,004 K		K	K
svchost.exe	2232						4,348 K	9,476 K		K	K
svchost.exe	2280						1,692 K	11,044 K		K	K
svchost.exe	2760						1,820 K	5,136 K		K	K
msdtc.exe	3160						1,040 K	8,604 K		K	K
taskhost.exe	3568		Host Process				9,044 K	15,884 K		200 K	1,380 K
SearchIndexer.exe	3828						10,244 K	25,984 K		K	K
svchost.exe	3308						3,000 K	6,020 K		K	K
taskhost.exe	3492						8,480 K	10,344 K		200 K	1,380 K
NisSrv.exe	4856						7,224 K	38,812 K		K	K
lsass.exe	560						7,016 K	11,864 K		K	K
lsm.exe	568						1,748 K	4,440 K		K	K
csrss.exe	456						30,716 K	40,264 K		K	K
winlogon.exe	500						324 K	8,788 K		K	K
explorer.exe	3688		Windows Expl				32,384 K	2,371,264 K	2	200 K	1,380 K
msseces.exe	3896		Microsoft Secu				9,212 K	16,068 K		200 K	1,380 K
ONENOTEM.EXE	3968		Send to OneN				1,060 K	8,964 K		200 K	1,380 K
TSVNCache.exe	1380		TortoiseSVN s				1,368 K	8,064 K		200 K	1,380 K
POWERPNT.EXE	4144		Microsoft PowerPoint	Microsoft Corporation	4,784,188	947,940 K	229,288 K	484,584 K		200 K	1,380 K
FoxitReader.exe	3744		Foxit Reader 8.1	Foxit Software Inc.	456,215	366,548 K	62,408 K	87,744 K	29	200 K	1,380 K
FoxitReaderUpdater.exe	676		Foxit Updater	Foxit Corporation	26,138	123,664 K	12,300 K	97,804 K		200 K	1,380 K
procexp.exe	4768		Sysinternals Process Explorer	Sysinternals - www....	6,989	75,032 K	6,312 K	6,344 K		200 K	1,380 K
procexp64.exe	2412	0.76	Sysinternals Process Explorer	Sysinternals - www....	33,240	166,348 K	28,672 K	28,732 K	27	200 K	1,380 K
jusched.exe	3092		Java Update Scheduler	Oracle Corporation	9,444	111,948 K	4,356 K	15,004 K		200 K	1,380 K

procexp64.exe:2412 Properties

TCP/IP | Security | Environment | Strings

Image | Performance | Performance Graph | Threads

CPU

Priority	13
Kernel Time	0:00:08.143
User Time	0:00:03.728
Total Time	0:00:11.871
Cycles	22,277,696,603

Virtual Memory

Private Bytes	17,028 K
Peak Private Bytes	17,096 K
Virtual Size	166,348 K
Page Faults	33,267
Page Fault Delta	27

Physical Memory

Memory Priority	5
Working Set	28,672 K
WS Private	14,232 K
WS Shareable	14,440 K
WS Shared	11,260 K
Peak Working Set	28,732 K

I/O

I/O Priority	Normal
Reads	58
Read Delta	
Read Bytes Delta	0
Writes	3
Write Delta	
Write Bytes Delta	0
Other	6,864
Other Delta	
Other Bytes Delta	0

Handles

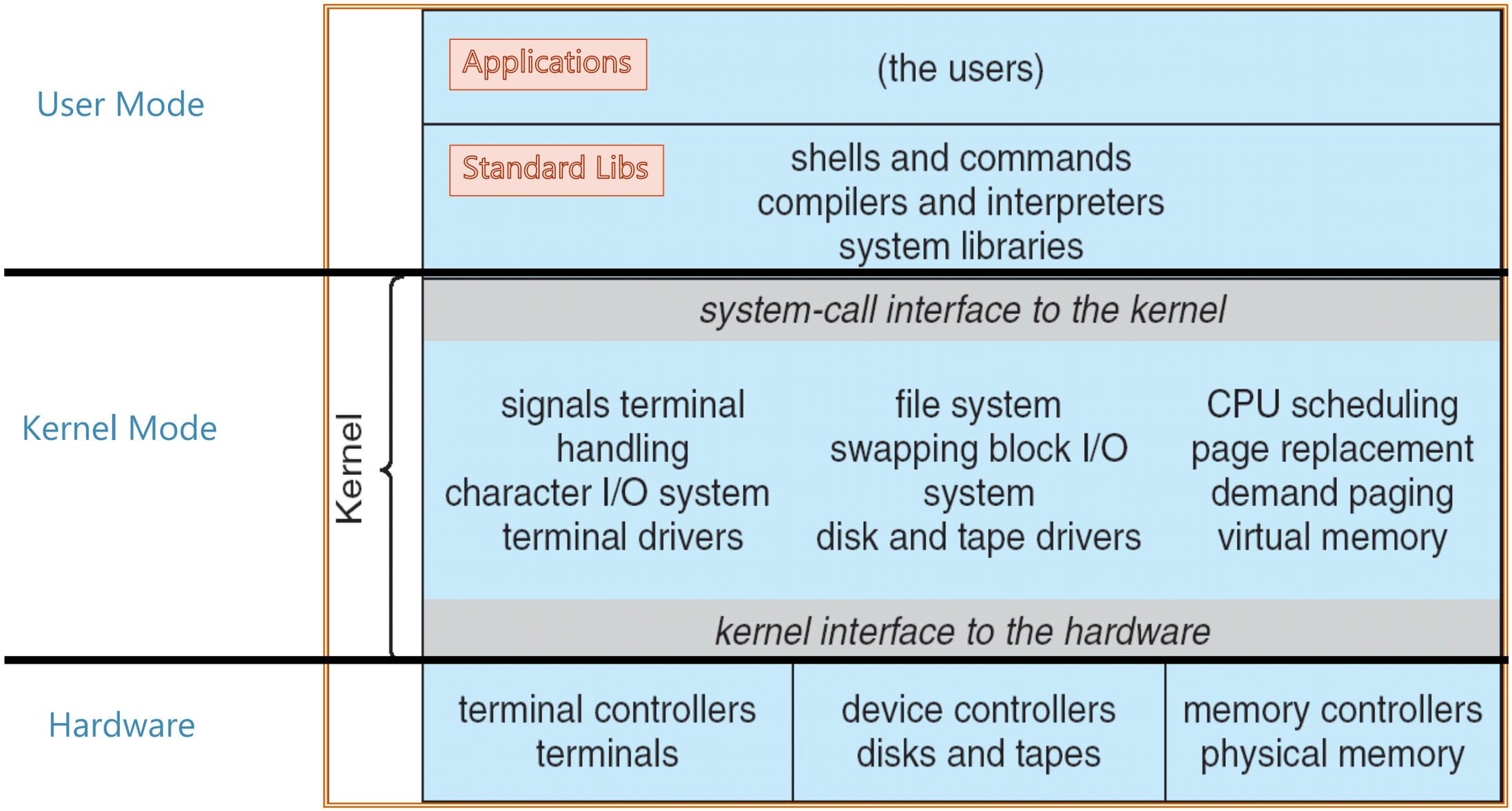
Handles	315
GDI Handles	214
USER Handles	216

OK Cancel

CPU Usage: 0.76% | Commit Charge: 29.41% | Processes: 62 | Physical Usage: 46.83%

کتابخانه‌های سیستمی

- چرا در زمان استفاده از توابع متوجه توابع سیستمی نمی‌شویم؟
  - توابع سیستمی که موجب فراخوانی سیستمی می‌شوند در کتابخانه‌های سیستمی تعبیه شده‌اند
  - عملا زمانی که ما تابع کتابخانه را صدا میزنیم این تابع در ادامه کارش تابع سیستمی معادل خودش را صدا میزند.



کتابخانه‌های سیستمی

- استاندارد POSIX (Portable Operating System Interface) یک استاندارد واحد برای کتابخانه‌های سیستمی تعریف می‌کند.
- در حقیقت سیستم عاملهای سازگار با استاندارد POSIX یک مجموعه از APIها را برای کار با سیستم عامل معرفی می‌کنند.
- بدین ترتیب کدهای نوشته شده در تمامی سیستم عاملهای سازگار قابل اجراست (Portable Code)

کتابخانه‌های سیستمی