فصل ۳: پردازه ها

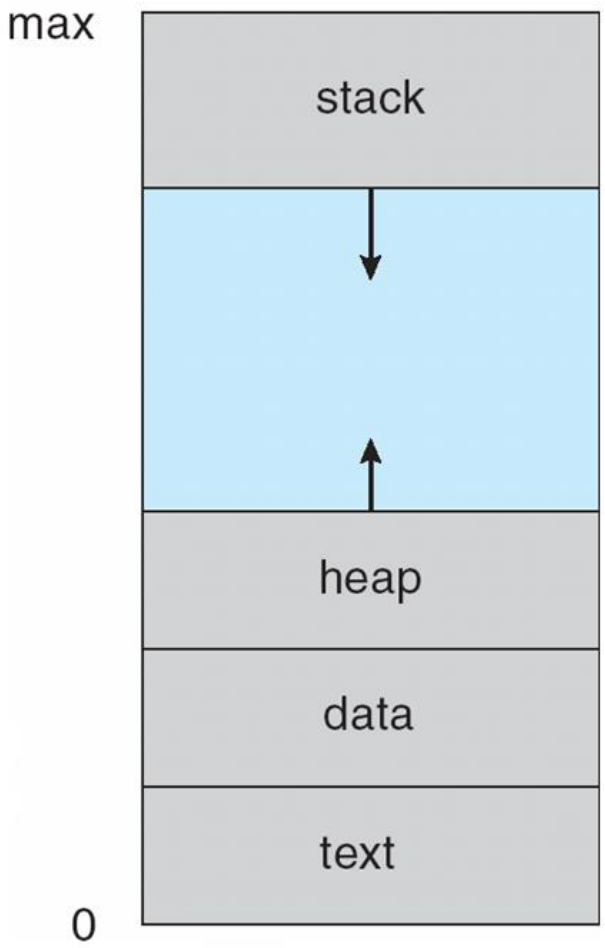
فصل ۳: پردازه ها

- مفهوم پردازه (Process)
- برنامه ریزی پردازه ها
- عملگرهای روی پردازه ها
- ارتباطات بین پردازه ای و نمونه های آن
- ارتباط ها در سامانه های Client-Server

مفهوم پردازه

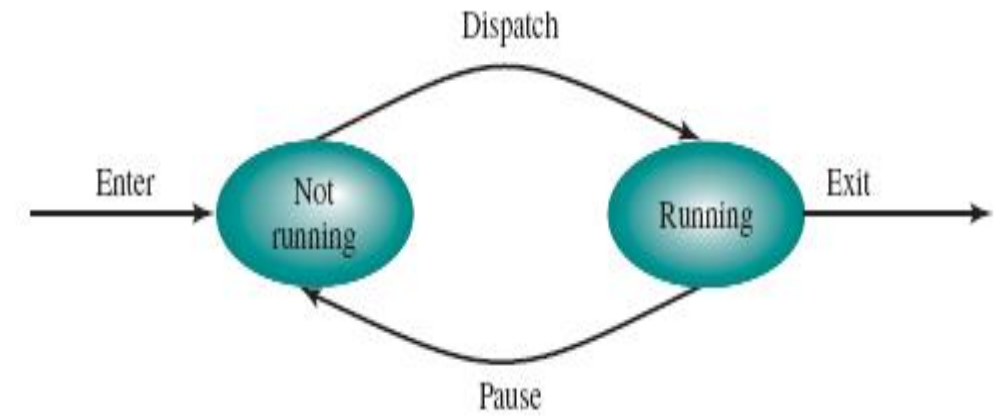
- سیستم عامل ها برنامه های مختلفی را اجرا می کنند.
  - سیستم عامل های **Batch** نام **jobs** را بکار می برند
  - سیستم عامل های **اشتراک زمانی** نام **user programs** ویا **tasks** را به کار می برند.
- در کتاب های مرجع سیستم عامل معمولا این دو نام به جای هم به کار می روند.
- **پردازه** : یک برنامه در حال اجرا به همراه منابع در اختیار آن
- بخش های مختلف یک پردازه
  - کد برنامه (بخش متن یا text نیز نامیده می شود.)
  - وضعیت فعلی پردازه شامل ثبات ها و آدرس فعلی PC
  - پشته که اطلاعات موقت را ذخیره کرده است.
  - بخش داده که متغیرهای عمومی را ذخیره کرده است.
  - بخش Heap که شامل حافظه تخصیص داده شده به صورت پویا است.

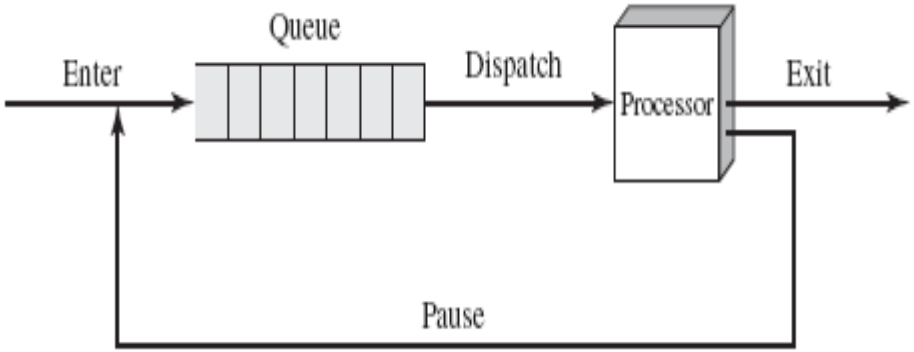
پردازه درون حافظه



وضعیت پردازه (پردازه با دو وضعیت)

- همزمان با اجرای پردازه وضعیت آن پردازه نیز تغییر می کند.
- وضعیت های پردازه
  - آماده اجرا
  - در حال اجرا

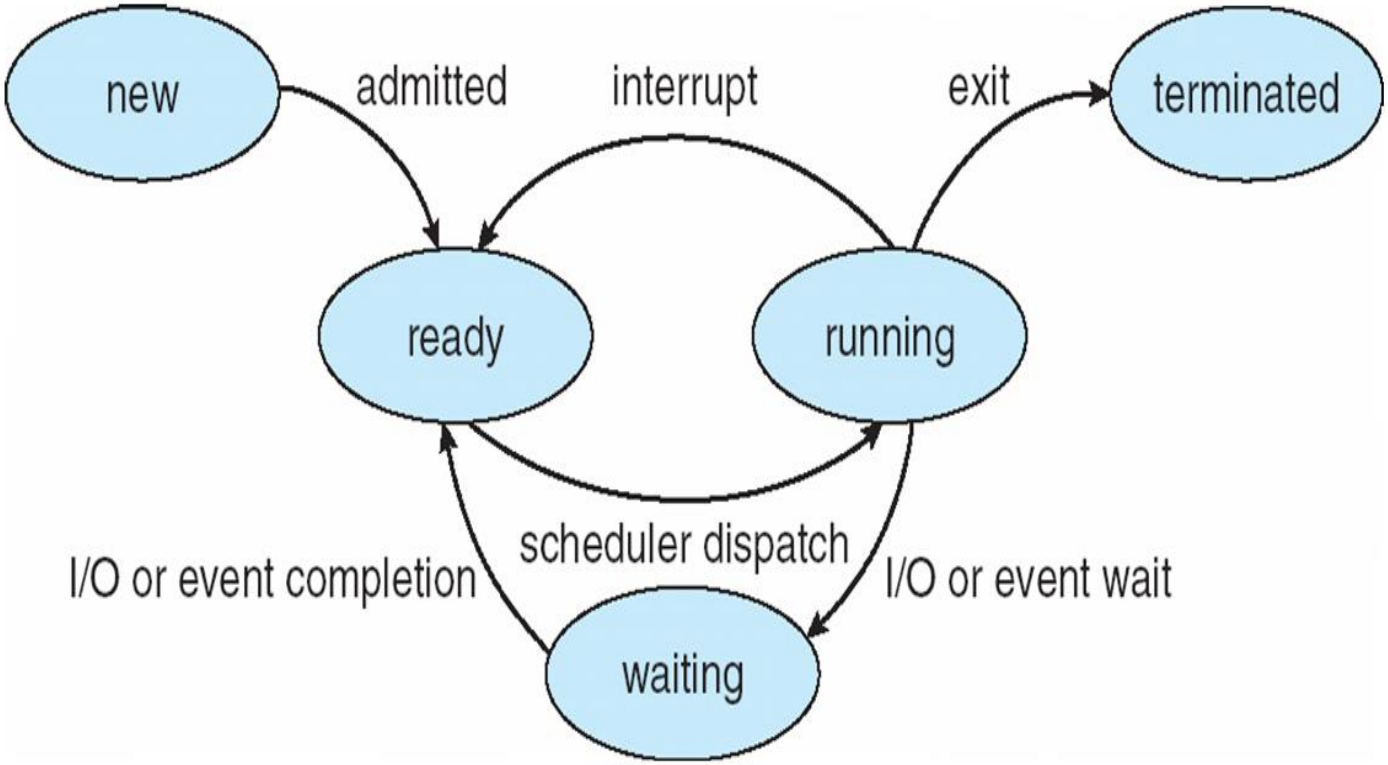




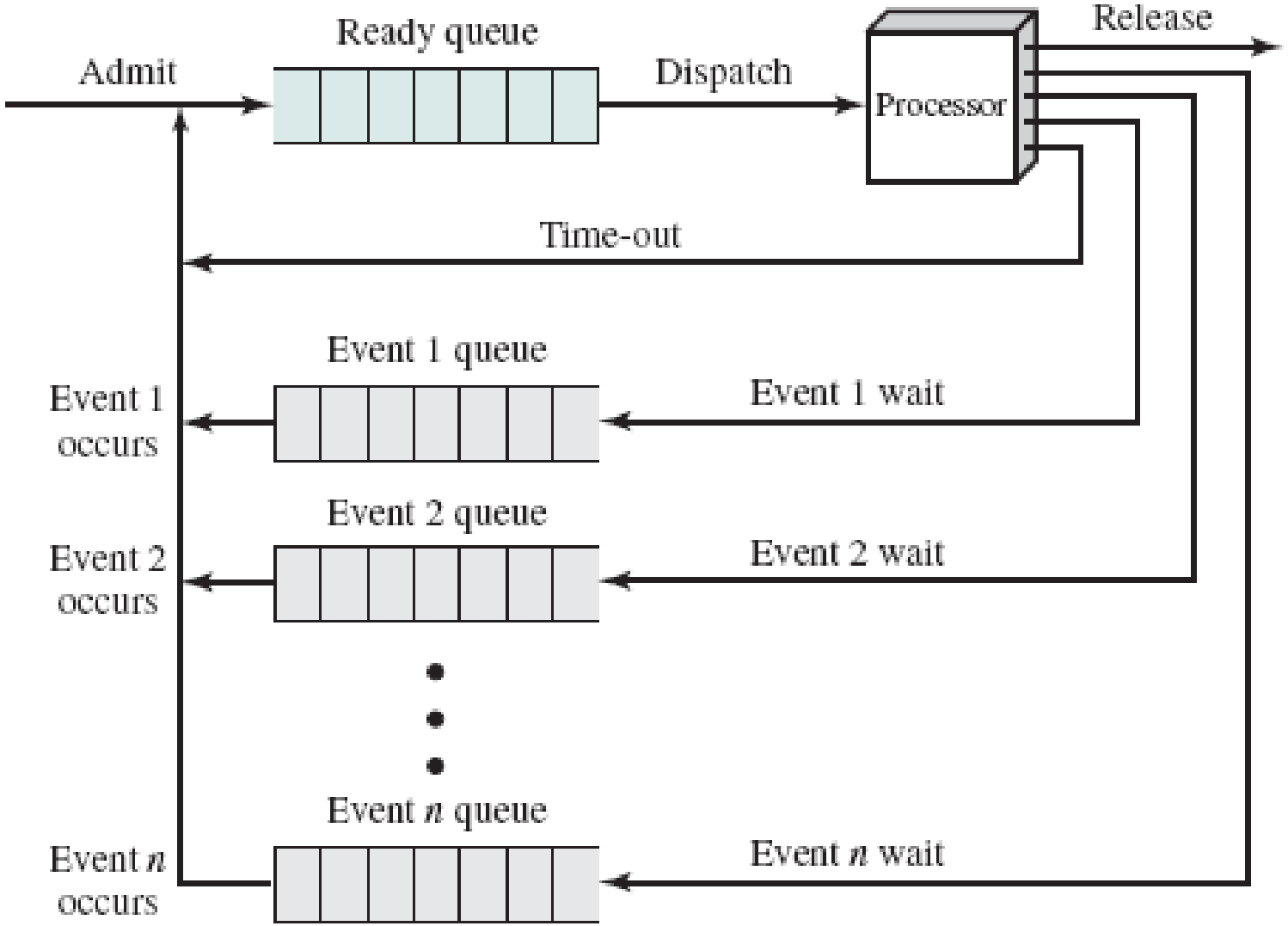
وضعیت پردازه (پردازه با پنج وضعیت)

- همزمان با اجرای پردازه وضعیت آن پردازه نیز تغییر می کند.
- وضعیت های پردازه
  - جدید (New)
  - در حال اجرا (Running)
  - در حال انتظار یا مسدود (Waiting or Blocked)
  - آماده اجرا (Ready)
  - پایان یافته (Terminated)

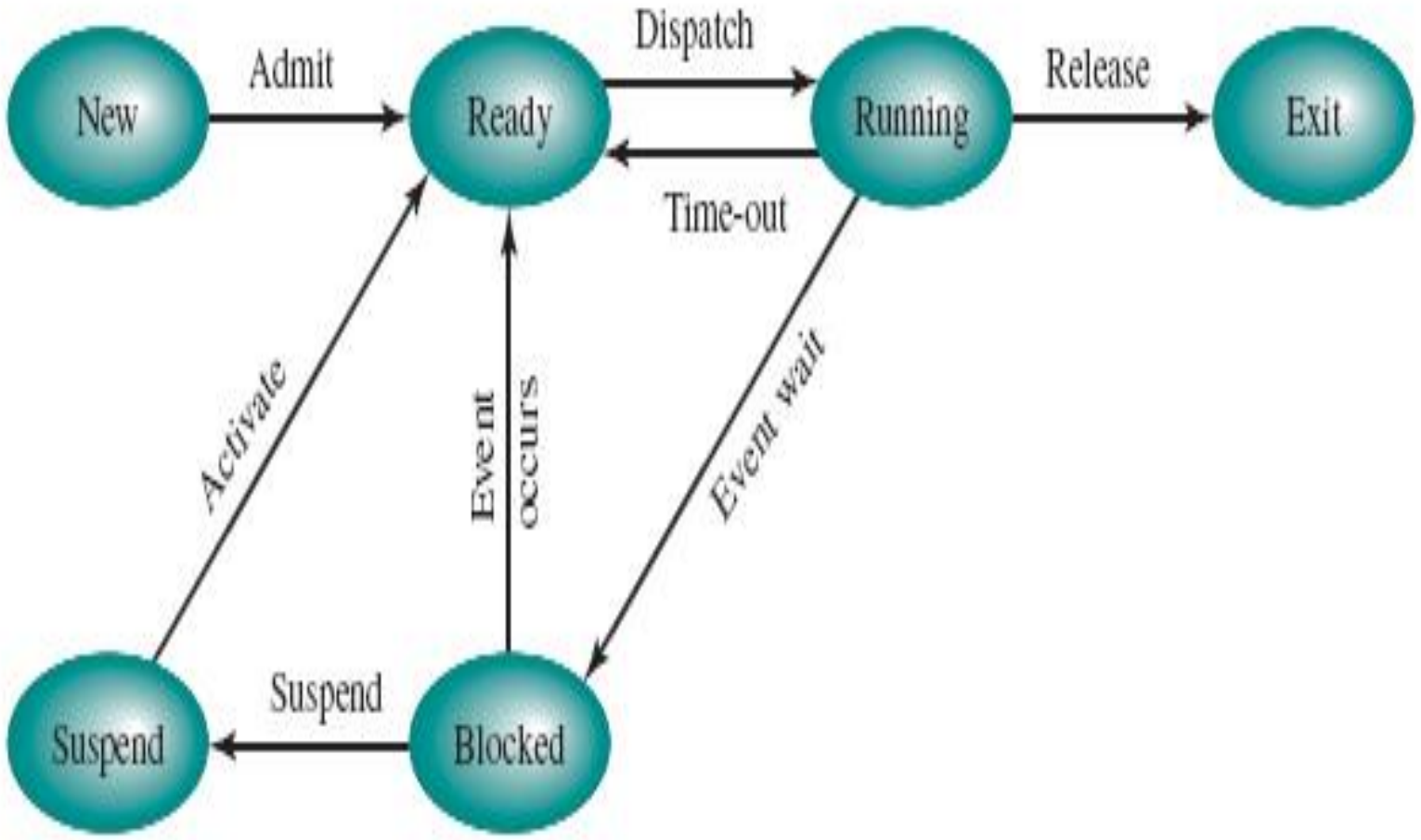
نمودار تغییر وضعیت پردازه



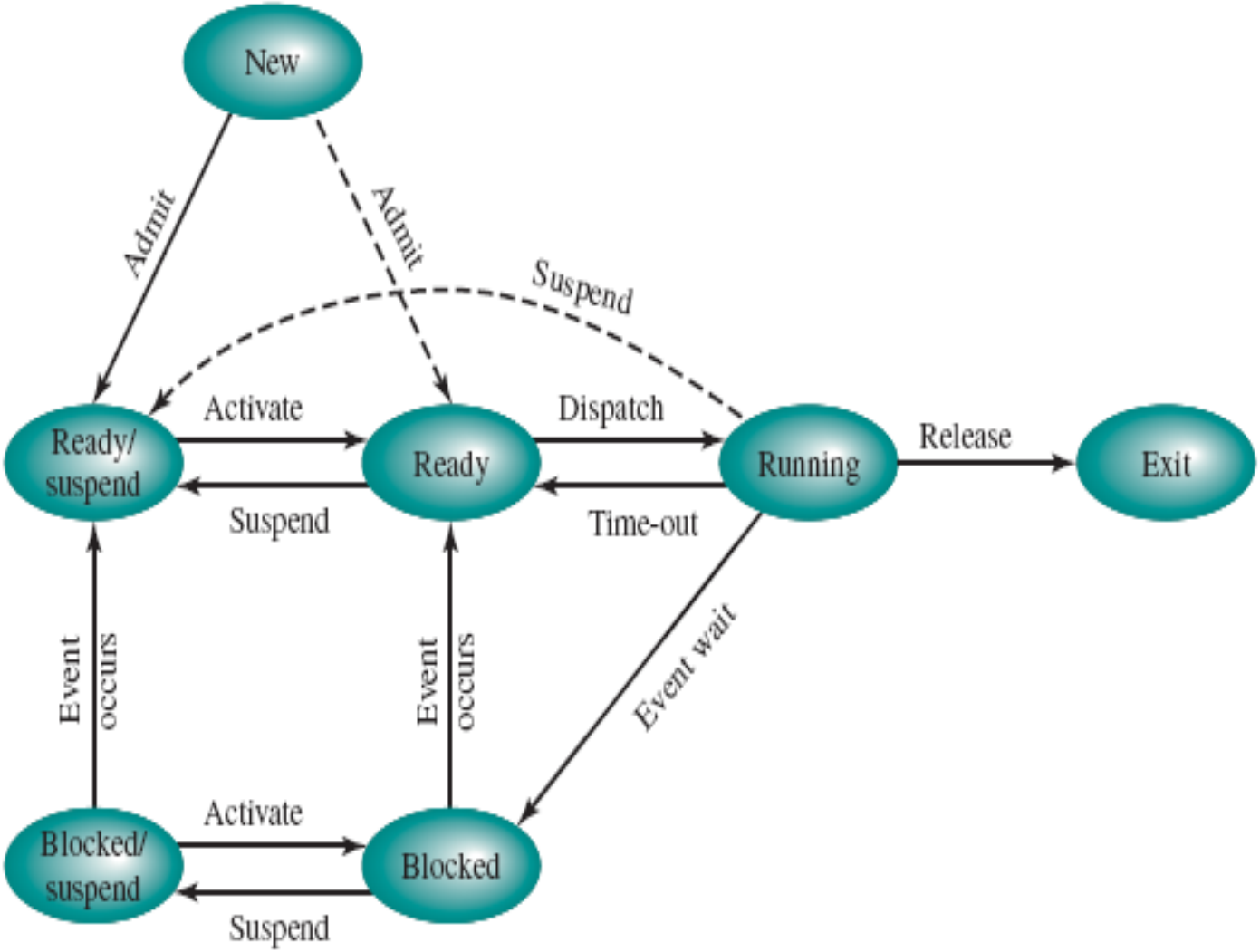
یک مدل صف برای پردازه با پنج وضعیت



نمودار تغییر وضعیت برای پردازه با شش وضعیت

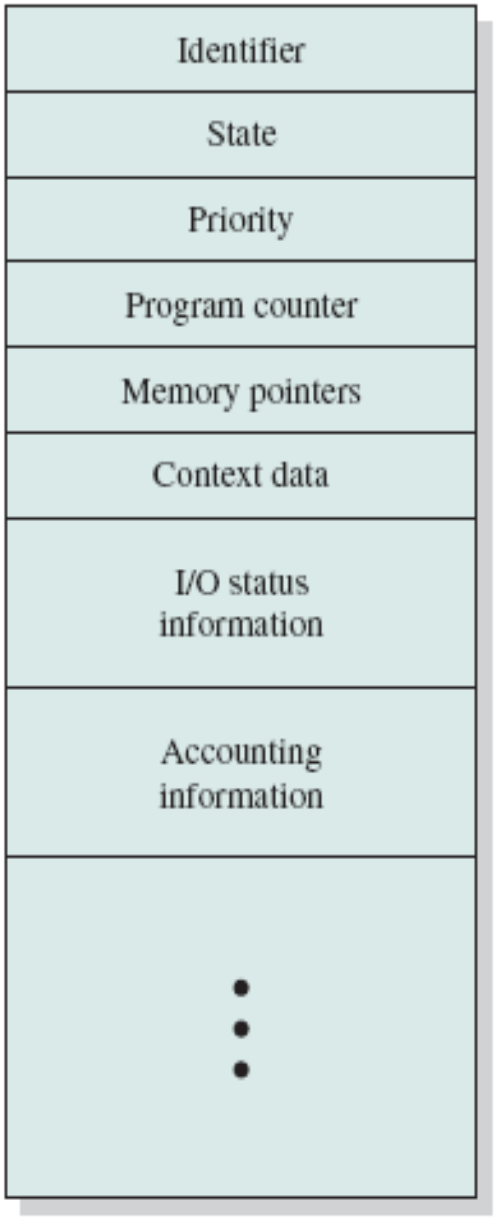


نمودار تغییر وضعیت برای پردازه با هفت وضعیت

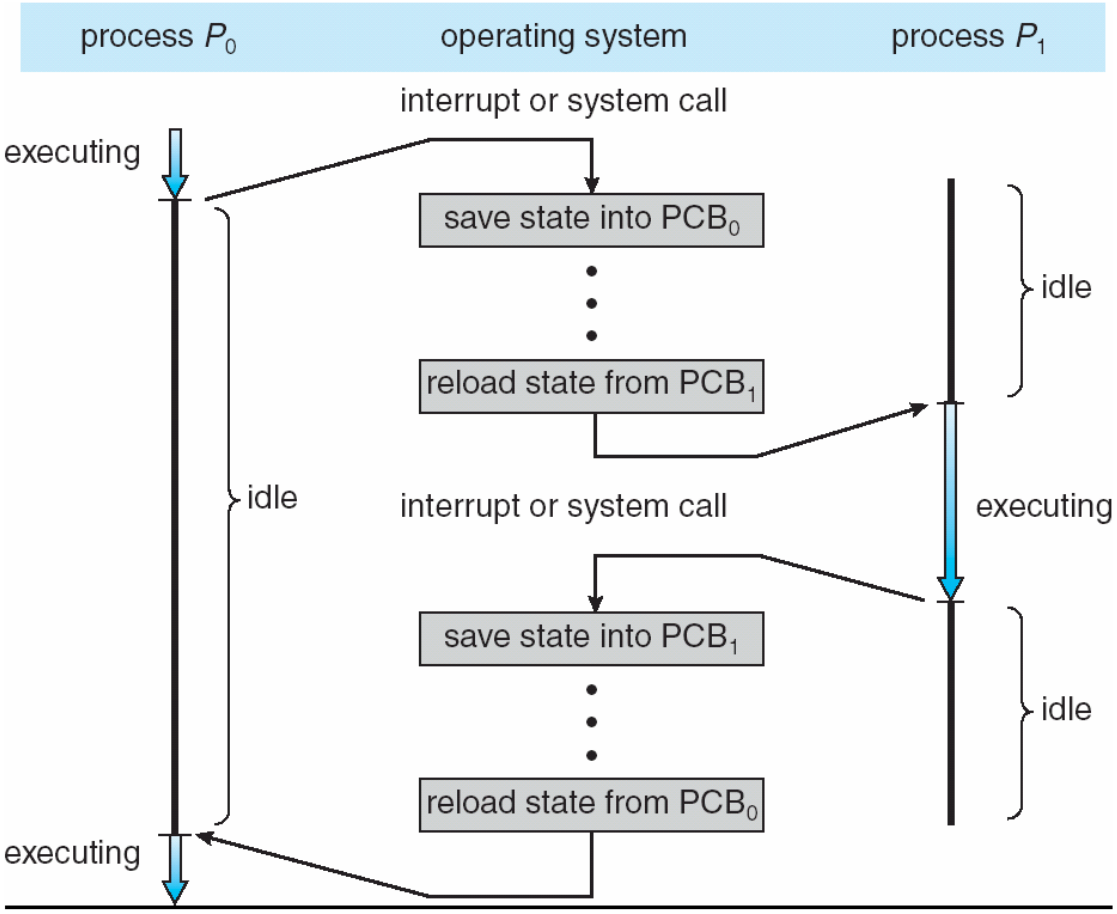


ساختمان داده لازم برای نشان دادن پردازه ها

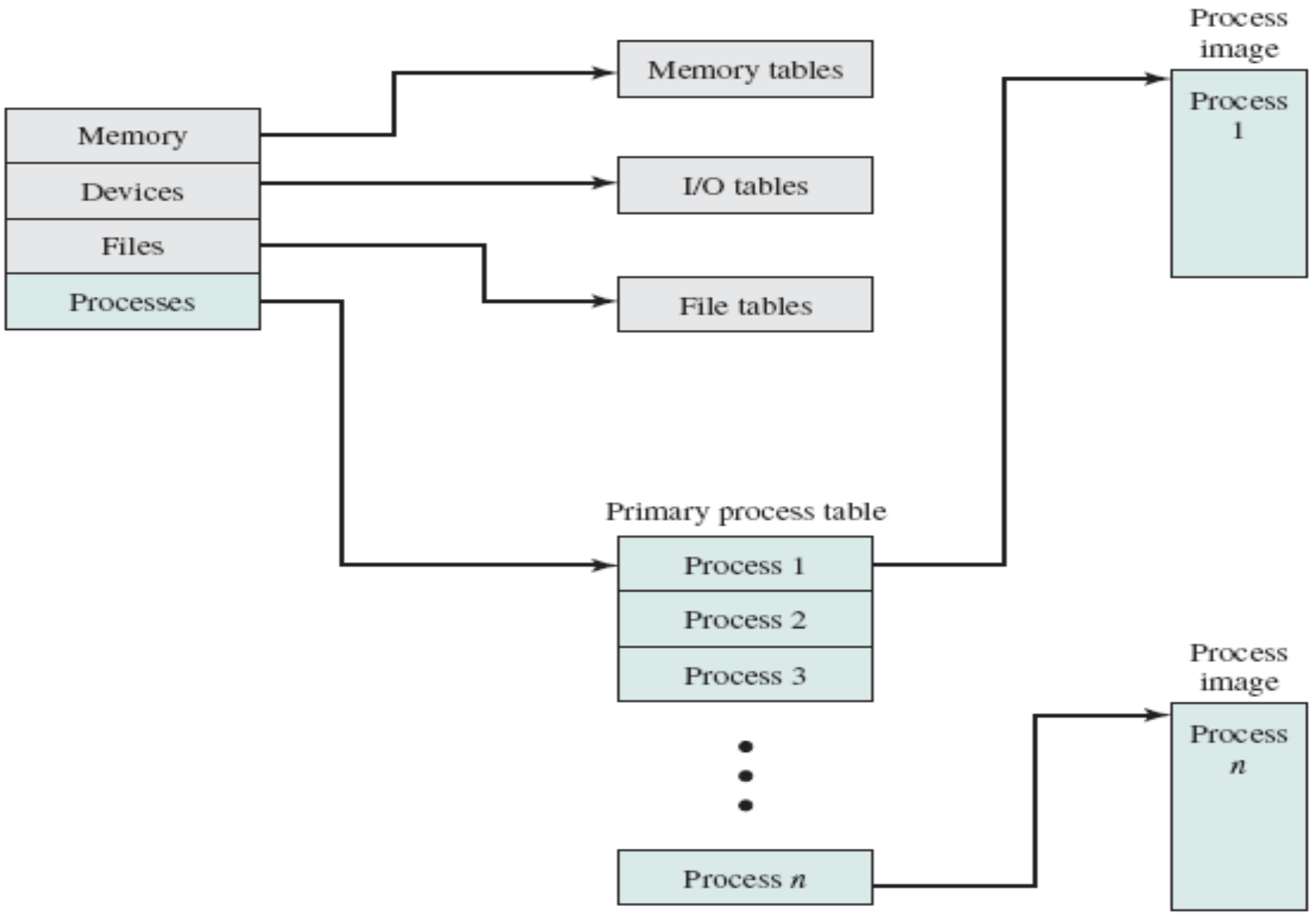
- مشخصات هر پردازه در سیستم عامل با یک ساختمان داده به نام **Process Control Block (PCB)** یا **Task Control Block (TCB)** نشان داده می شود.



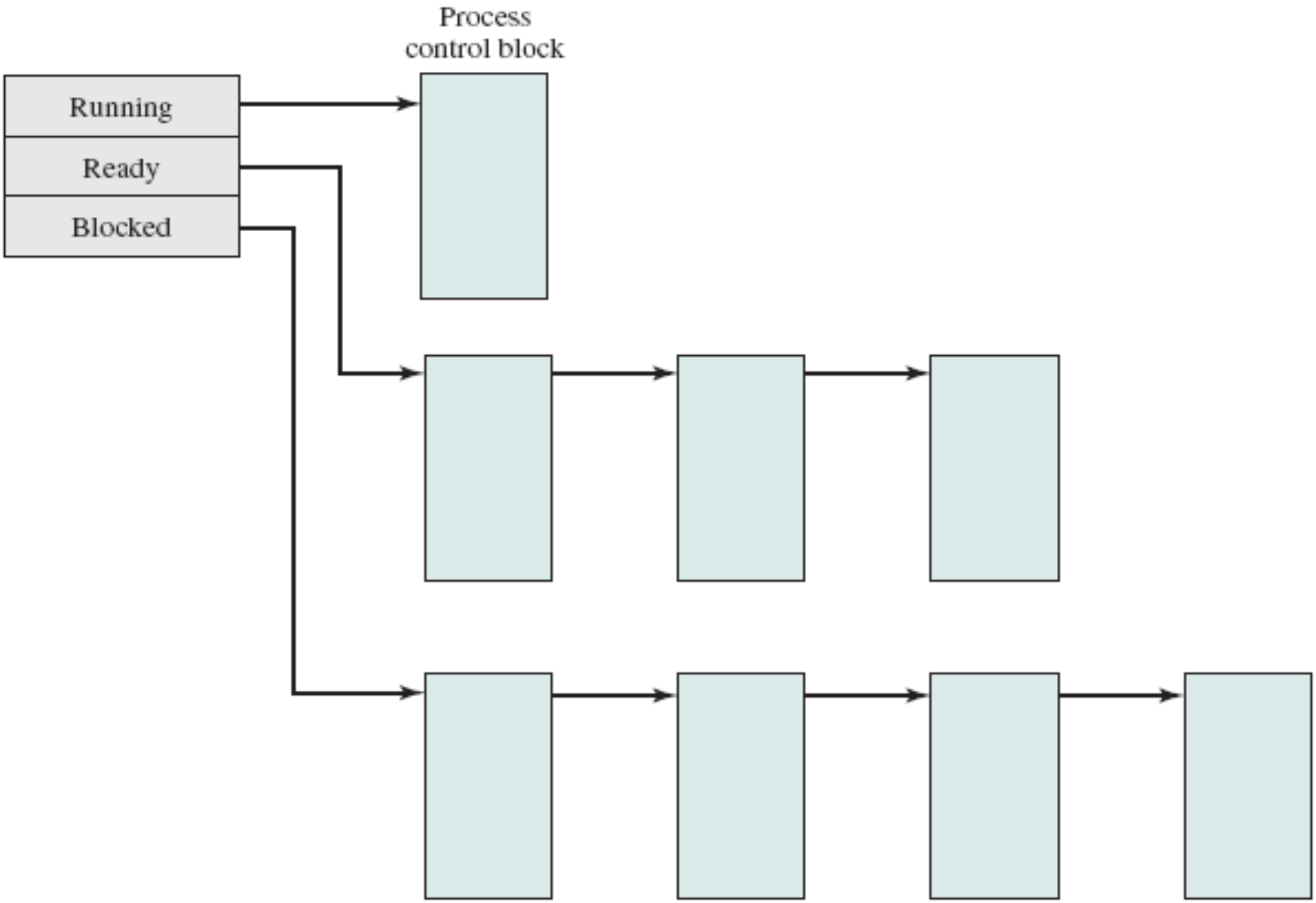
شیوه تغییر پردازه در حال اجرا



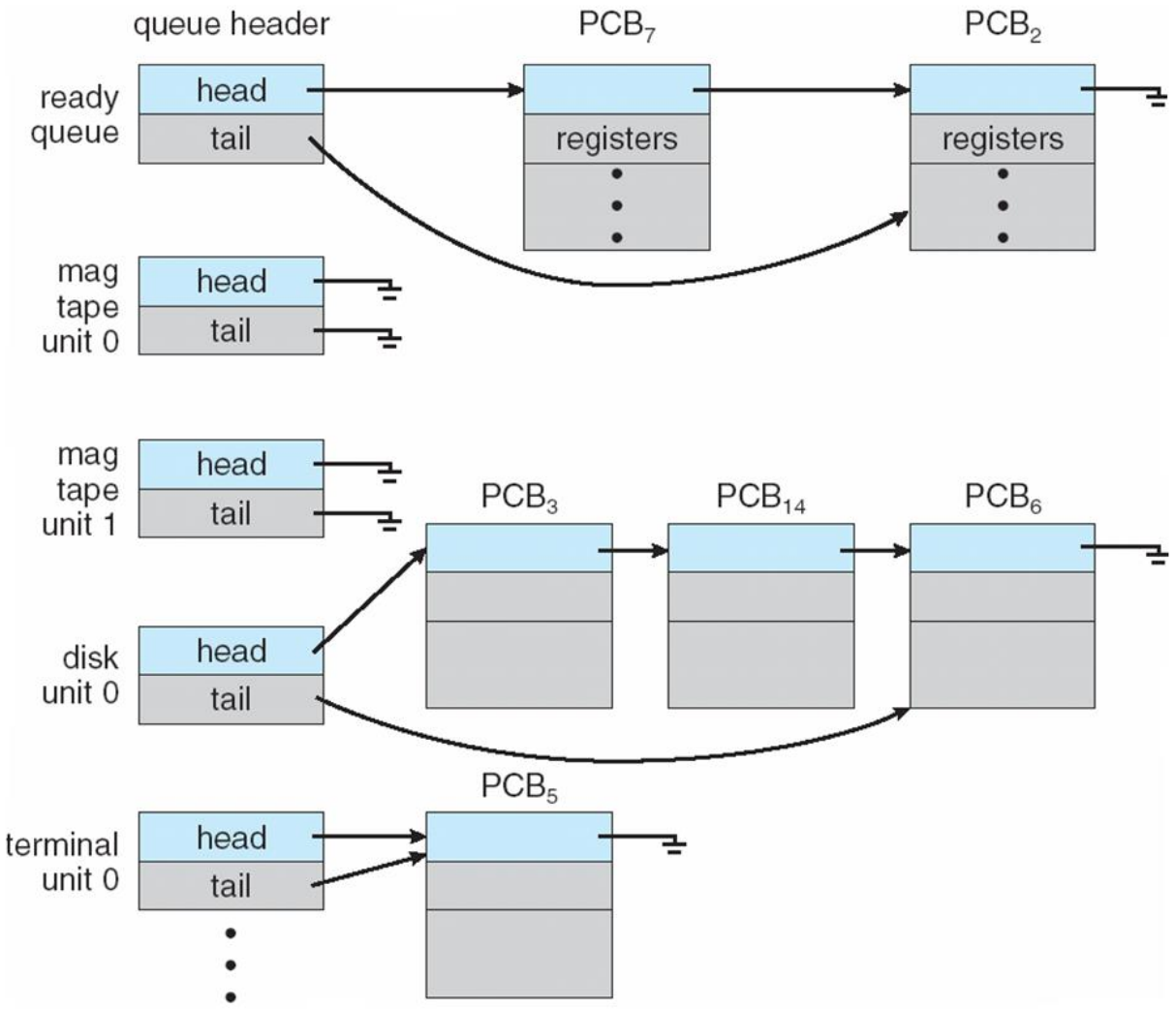
ساختار عمومی جداول کنترلی سیستم عامل



انواع صف ها در سیستم عامل



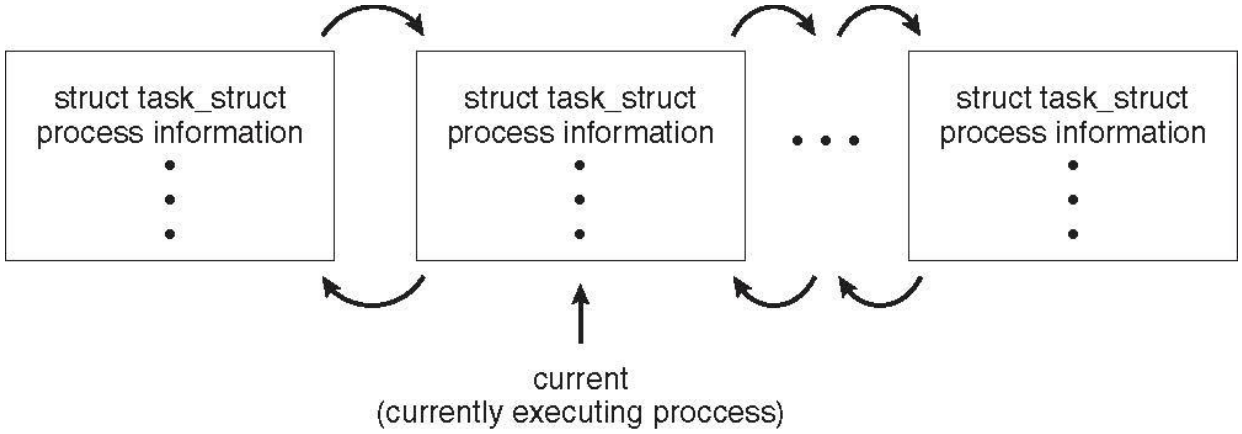
انواع صف ها در سیستم عامل



نمایش یک پردازه در لینوکس

با ساختار `task_struct` در لینوکس نشان داده می شود.

```
pid t_pid; /* process identifier */
long state; /* state of the process */
unsigned int time_slice /* scheduling information */
struct task_struct *parent; /* this process's parent */
struct list_head children; /* this process's children */
struct files_struct *files; /* list of open files */
struct mm_struct *mm; /* address space of this process */
```

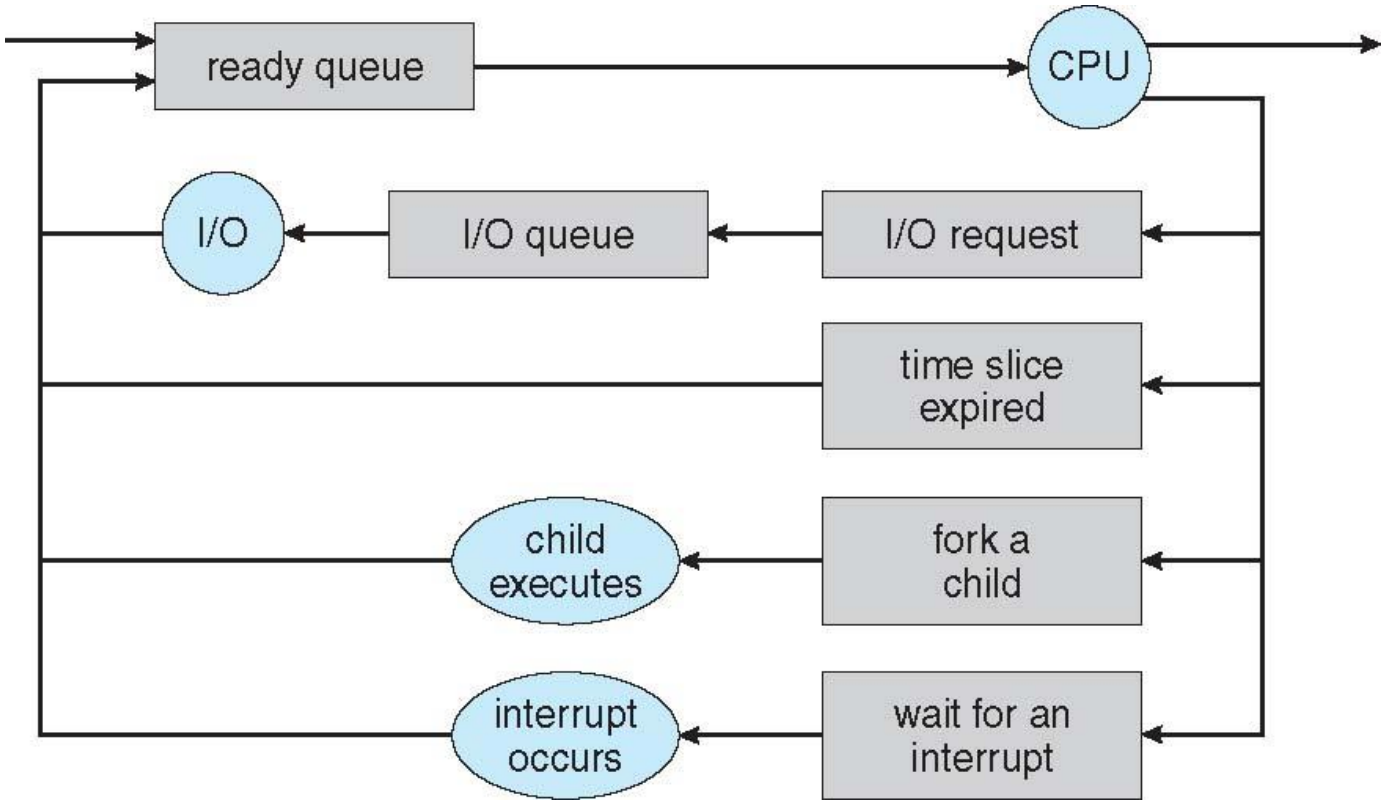


نخ یا ریسه

- بیشتر سیستم عامل ها اجازه می دهند یک پردازه حاوی بیش از یک ریسه (**Thread**) باشد.
- نمونه
  - بار گذاری چند تصویر در مرورگر وب
  - کنترل غلط املایی و تایپ در ویرایشگر متن مانند Word
- اگر یک سیستم عامل اجازه می دهد که یک پردازه دارای بیش از یک ریسه باشد باید PCB امکانات لازم برای ذخیره سازی مشخصات چند ریسه را فراهم نماید.

برنامه ریزی پردازه ها

- نمودار صف بندی نشان دهنده صف ها، منابع و گردش کار در سیستم عامل است.



انواع برنامه ریزها در سیستم عامل

- یک پردازه در جرخه زندگی خود بین صف های برنامه ریزی مختلف مهاجرت می کند و سیستم عامل براساس راهبردهایی از این صف ها پردازه ها را انتخاب می کند. هر صف برنامه ریزی ویژه خود را دارد.
- در سیستم عامل ها سه نوع برنامه ریز وجود دارد.
  - برنامه ریز بلند مدت
  - برنامه ریز میان مدت
  - برنامه ریز کوتاه مدت (در خیلی از سیستم عامل ها این تنها برنامه ریز سیستم عامل است)

- برنامه ریز کوتاه مدت
  - برنامه ریز کوتاه مدت (یا برنامه ریز پردازنده) پردازه ای را انتخاب می نماید که می بایست بر روی پردازنده اجرا شود و سپس پردازنده را به این پردازه تخصیص می دهد.
  - تعداد دفعات اجرای این برنامه ریز خیلی زیاد است و ممکن است این برنامه ریز در هر چند میلی ثانیه اجرا شود. بنابراین باید بسیار سریع باشد.

- برنامه ریز بلند مدت
  - این برنامه ریز مشخص می کند که کدام پردازه باید در صف پردازه های آماده اجرا قرار گیرند.
  - تعداد دفعات اجرای این برنامه ریز کم است و ممکن است در هر چند ثانیه یا دقیقه اجرا شود.
  - این برنامه ریز درجه چند برنامگی سیستم را کنترل می کند.

انواع برنامه ریزها در سیستم عامل

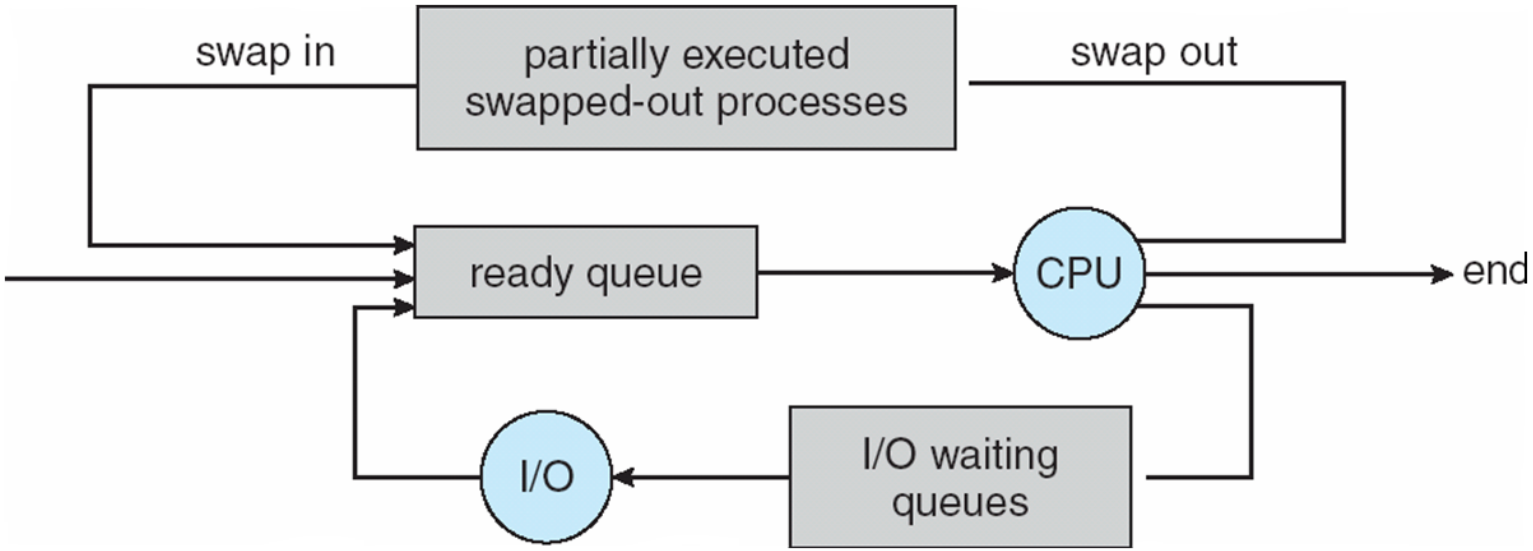
- یکی اهداف برنامه ریز بلند مدت انتخاب مناسب پردازه ها جهت اجرا است تا بهره وری سیستم افزایش یابد.
- انواع پردازه ها در سیستم
  - پردازه های IO-Bound
  - پردازه های CPU-Bound

- یک الگوریتم زمان بندی بلند مدت خوب باید ترکیب مناسبی از پردازه ها را برای اجرا انتخاب نماید تا بهروه وری سامانه افزایش یابد.

- برنامه ریز بلند مدت بیشتر در سیستم عامل های Batch استفاده می شد و در سیستم عامل های اشتراک زمانی مانند Windows و Unix وجود ندارد. بنابراین پایداری این نوع سیستم ها به میزان بسیار زیادی به محدودیت هایی که بصورت فیزیکی روی سیستم گذاشته می شود مانند تعداد کاربران, تعداد پردازه ها و ... وابسته است.

برنامه ریز میان مدت

- برخی از سیستم عامل ها مانند برخی از سیستم عامل های اشتراک زمانی از برنامه ریز میان مدت استفاده می کنند.
- این برنامه ریز درجه چند برنامگی سیستم را کنترل می کند. هر وقت درجه چند برنامگی سیستم از یک آستاده گذشت برخی از پردازه ها به حالت تعلیق در آمده و به دستگاه جانبی منتقل می شوند و در هنگامی که بار سیستم کاهش پیدا نمود دوباره به حال فعال برگردانده می شوند.



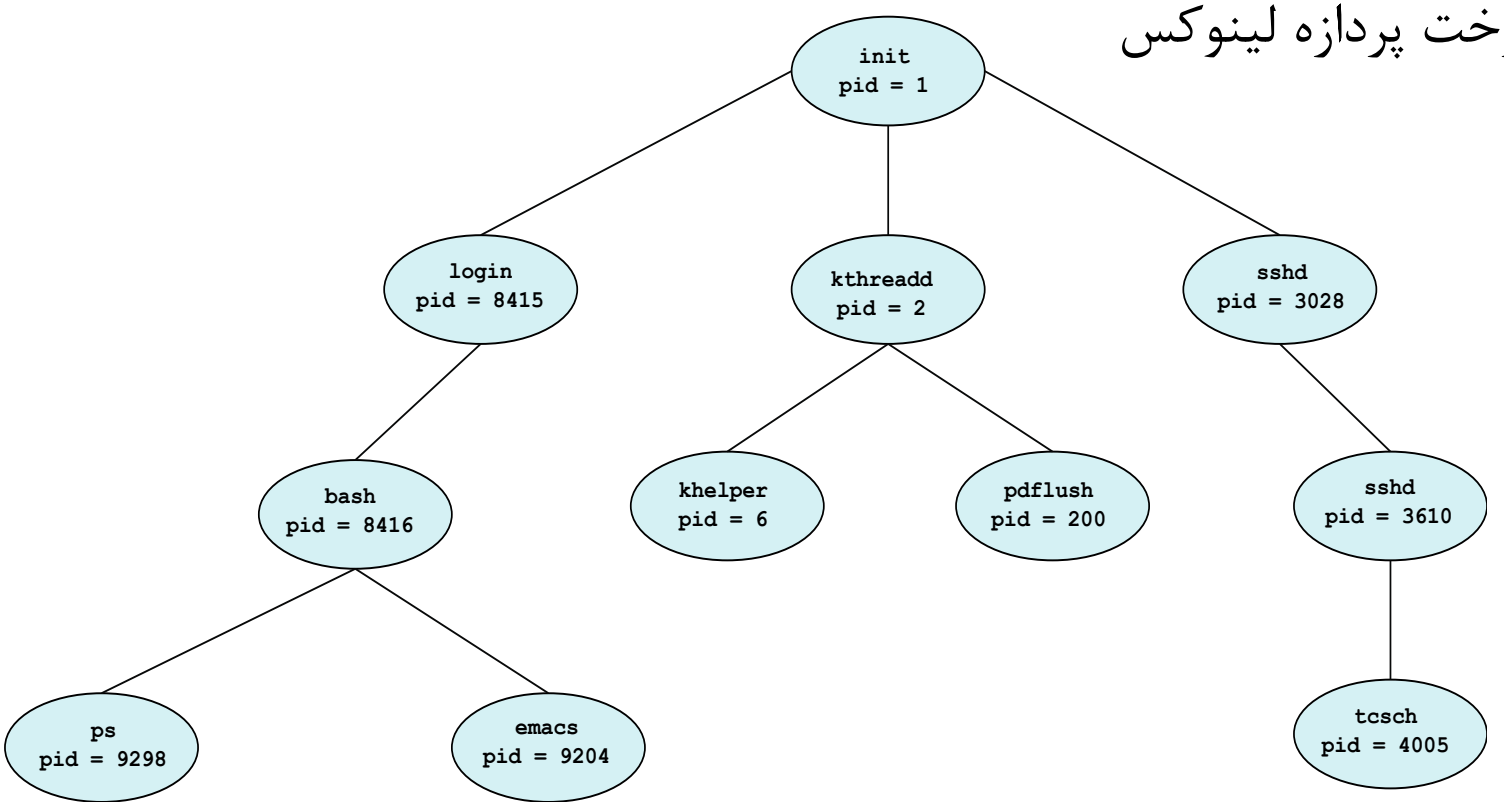
تعویض زمینه (Context Switch)

- هنگامی که پردازنده از یک پردازه گرفته می شود وضعیت آن پردازه باید ذخیره شود و هنگامی که پردازنده در اختیار که پردازه قرار می گیرد وضعیت پردازه می بایست دوباره بارگذاری شود.

- این وضعیت درون PCB ذخیره میشود.
- تعویض زمینه سربار است و در این مدت سیستم کار مفیدی انجام نمی دهد.
- زمان تعویض زمینه به سخت افزار وابسته است و تا چند میلی ثانیه طول می کشد.
- برخی از سخت افزارها دستورهای ویژه ای برای این منظور دارند که وضعیت را یک باره ذخیره یا بارگذاری می نماید.
- برخی از سخت افزارها مانند UltraSparc چند مجموعه ثبات دارند

عملگرهای روی پردازه ها

- سیستم باید عملگرهایی مانند
  - ایجاد پردازه
  - پایان پردازه
  - درخت پردازه لینوکس



ایجاد پردازه در یونیکس

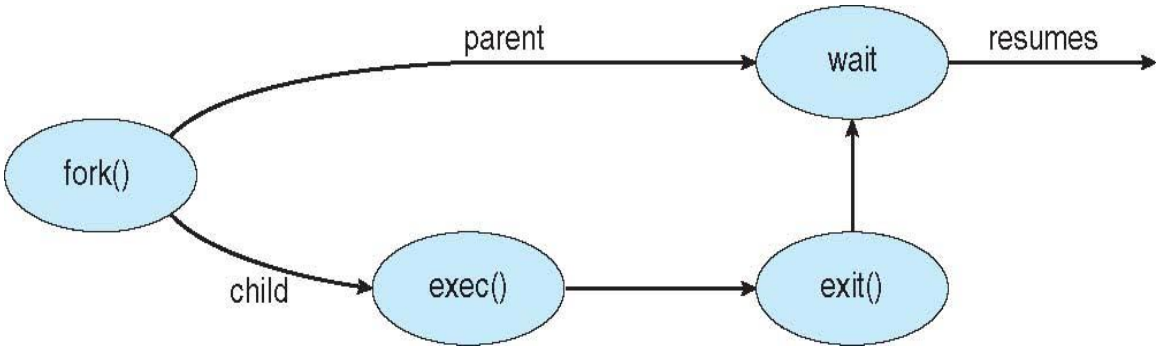
```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>

int main()
{
pid_t pid;

    /* fork a child process */
    pid = fork();

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

    return 0;
}
```



ایجاد پردازه در ویندوز

```
#include <stdio.h>
#include <windows.h>

int main(VOID)
{
STARTUPINFO si;
PROCESS_INFORMATION pi;

    /* allocate memory */
    ZeroMemory(&si, sizeof(si));
    si.cb = sizeof(si);
    ZeroMemory(&pi, sizeof(pi));

    /* create child process */
    if (!CreateProcess(NULL, /* use command line */
      "C:\\WINDOWS\\system32\\mspaint.exe", /* command */
      NULL, /* don't inherit process handle */
      NULL, /* don't inherit thread handle */
      FALSE, /* disable handle inheritance */
      0, /* no creation flags */
      NULL, /* use parent's environment block */
      NULL, /* use parent's existing directory */
      &si,
      &pi))
    {
       fprintf(stderr, "Create Process Failed");
       return -1;
    }
    /* parent will wait for the child to complete */
    WaitForSingleObject(pi.hProcess, INFINITE);
    printf("Child Complete");

    /* close handles */
    CloseHandle(pi.hProcess);
    CloseHandle(pi.hThread);
}
```

پایان اجرای پردازه

- هرگاه پردازه ای بخواهد به کار خود پایان دهد فراخوان سیستمی exit() را صدا میزند.
- در این زمان پردازه وضعیت خود را به پدر خود (از طریق wait()) اطلاع میدهد.
- همه منابع این پردازه آزاد می شود.
- در حالت های دیگری هم کار یک پردازه پایان می یابد.
  - پدر یک پردازه براساس دلایلی مانند دلایل زیر با فراخوان abort() به کار فرزند پایان می دهد.
    - به فعالیت پردازه فرزند دیگر نیازی نیست.
    - فرزند بیش از حد منابع را مصرف نموده است.
    - کار پدر پایان یافته است و فرزند دیگر نمی تواند اجرا شود.

پایان جلسه چهارم

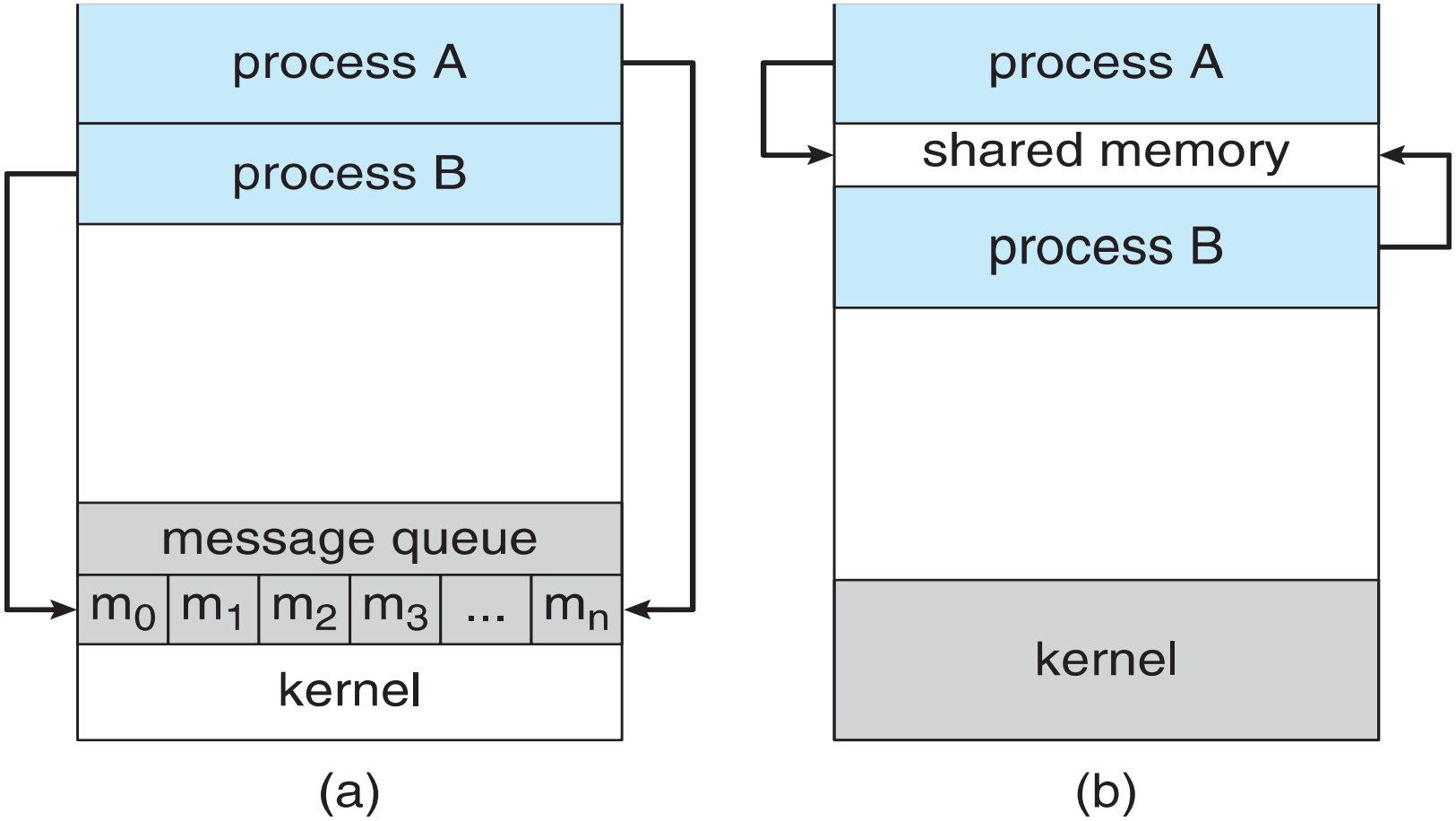
ارتباطات بین پردازه ای

- انواع پردازه های درون سیستم :
  - همکار :
    - پردازه هایی که روی هم تاثیر می گذارند یا از هم تاثیر می گیرند.
    - برخی از دلایل نیاز به پردازه های همکار
      - اشتراک اطلاعات
      - افزایش سرعت محاسبات (در صورت وجود چند هسته پردازشی)
      - پیمانه ای بودن سامانه
  - مستقل
- پردازه های همکار نیاز به ارتباطات بین پردازه ای دارند
- انواع ارتباطات بین پردازه ای : حافظه اشتراکی و ارتباط با پیام

مدل های ارتباطی بین پردازه ای

(a) ارتباط با پیام.

(b) حافظه اشتراکی.



مساله تولید کننده-مصرف کننده

- این مثال یک تعریف از پردازه هایی ارائه می نماید که یک یا چند پردازه اطلاعات را تولید و یک یا چند پردازه اطلاعات را مصرف می کنند.
- انواع مساله های تولید کننده-مصرف کننده
  - بافر نامحدود
  - بافر محدود (اطلاعات زیر بین پردازه ها به اشتراک گذاشته می شود)

```
#define BUFFER_SIZE 10
typedef struct {
  . . .
} item;

item buffer[BUFFER_SIZE];
int in = 0;
int out = 0;
```

- راه حل در صورتی که تنها از BUFFER_SIZE-1 عنصر آرایه استفاده شود درست است.

تولید کننده و مصرف کننده (بافر محدود)

تولید کننده

```
item next_produced;
while (true) {
/* produce an item in next produced */
      while (((in + 1) % BUFFER_SIZE) == out);
            /* do nothing */
       buffer[in] = next_produced;
       in = (in + 1) % BUFFER_SIZE;
}
```

مصرف کننده

```
item next_consumed;
while (true) {
    while (in == out)   ; /* do nothing */
     next_consumed = buffer[out];
     out = (out + 1) % BUFFER_SIZE;
     /* consume the item next consumed */
}
```

ارتباطات بین پردازه ای: حافظه اشتراکی

- یک ناحیه از حافظه بین دوپردازه ای که می خواهند با هم ارتباط داشته باشند به اشتراک گذاشته می شود.
- ارتباط بین پردازه ها تحت کنترل پردازه ها است نه سیستم عامل
- چالش اصلی ارائه یک سازوکار است که به پردازه های کاربر اجازه همگام شدن می دهد.
- همگام سازی در فصل ۵ شرح داده می شود.

ارتباط بین پردازه ها : تبادل پیام

- یک سازو کار برای ارتباط و همگام سازی پردازه ها
- پردازه ها بدون حافظه اشتراکی با هم تبادل اطلاعات می نمایند.
- در این شیوه دو عملگر زیر فراهم می شود.
  - **send**(*message*)
  - **receive**(*message*)
- اندازه پیام
  - طول ثابت
  - طول متغیر

ارتباط بین پردازه ها : تبادل پیام (ادامه)

- برای ارتباط دو پردازه P , Q می بایست مراحل زیر انجام شود.
  - برقراری یک خط ارتباطی بین آنها
  - فرستادن و دریافت پیام ها با کمک send/receive
- مسائل پیاده سازی
  - چگونگی برقراری خط ارتباطی
  - آیا خط ارتباطی را می توان برای بیش از دو پردازه بکار رود
  - طول پیام ثابت است یا متغیر
  - آیا خط ارتباطی یک طرفه است یا دو طرفه

ارتباط بین پردازه ها : تبادل پیام (ادامه)

- پیاده سازی خط ارتباطی
  - فیزیکی
    - حافظه اشتراکی
    - درگاه (Bus) سخت افزاری
    - شبکه
  - منطقی
    - جهت دار یا بدون جهت
    - همزمان یا نا همزمان
    - بافر خودکار یا بافر صریح

ارتباط بین پردازه ها : تبادل پیام (ارتباط مستقیم)

- پردازه ها باید صریحا نام همدیگر را بیاورند
  - **send** (*P*, *msg*) – send a msgto process P
  - **receive**(*Q*, *msg*) – receive a msgfrom process Q

- ویژگی های خط ارتباطی
  - ارتباط بصورت خودکار ایجاد می شود
  - ارتباط برای یک زوج پردازه به کار می رود
  - برای هر زوج تنها یک خط ارتباطی وجود دارد
  - خط می تواند یک طرفه اشد اما معمولا دوطرفه باید باشد

ارتباط بین پردازه ها : تبادل پیام (ارتباط غیرمستقیم)

- پیام ها از طریق صنوق پستی (Mailbox) یا درگاه (Port) دریافت و فرستاده می شود.
- هر صنوق پستی یک شناسه یکتا دارد.
- پردازه ها در صورتی می توانند با هم ارتباط داشته باشند که یک صندوق پستی داشته باشند.
- ویژگی های خط ارتباطی
  - ارتباط در صورتی برقرار می شود که صتدوق پستی مشترک داشته باشند.
  - ارتباط بین چند پردازه می تواند وجود داشته باشد.
  - هر زوج پردازه می توانند چندین خط ارتباطی مستقل با هم داشته باشند.
  - ارتباط می تواند یک طرفه یا دو طرفه باشد.

ارتباط بین پردازه ها : تبادل پیام (ارتباط غیرمستقیم)

- عملیات مورد نیاز
  - ایجاد صندوق پستی یا درگاه (در صورت نیاز)
  - فرستادن و دریافت پیام از طریق صندوق پستی یا درگاه
  - حذف صندوق پستی
- دستورها بصورت زیر می باشد.

send(*A*, *msg*) – send a msgto mailbox A

receive(*A*, *msg*) – receive a msgfrom mailbox A

همگام سازی

- تبادل پیام می تواند همزمان (Blocking) یا ناهمزمان باشد(Non-blocking).
- تبادل پیام همزمان
  - فرستادن همزمان
  - دریافت همزمان
- تبادل پیام ناهمزمان
  - فرستادن ناهمزمان
  - دریافت ناهمزمان (یا یک پیام دریافت می کند یا یک پیام تهی دریافت می نماید)

همگام سازی: تولید کننده–مصرف کننده

```
message next_produced;
while (true) { /* produce an item in next produced */
        send(next_produced);
}
```

```
message next_consumed;
while (true) {
   receive(next_consumed);
      /* consume the item in next consumed */
}
```

پایان جلسه پنجم