



# **Operating Systems**

## **Processes-Part2**

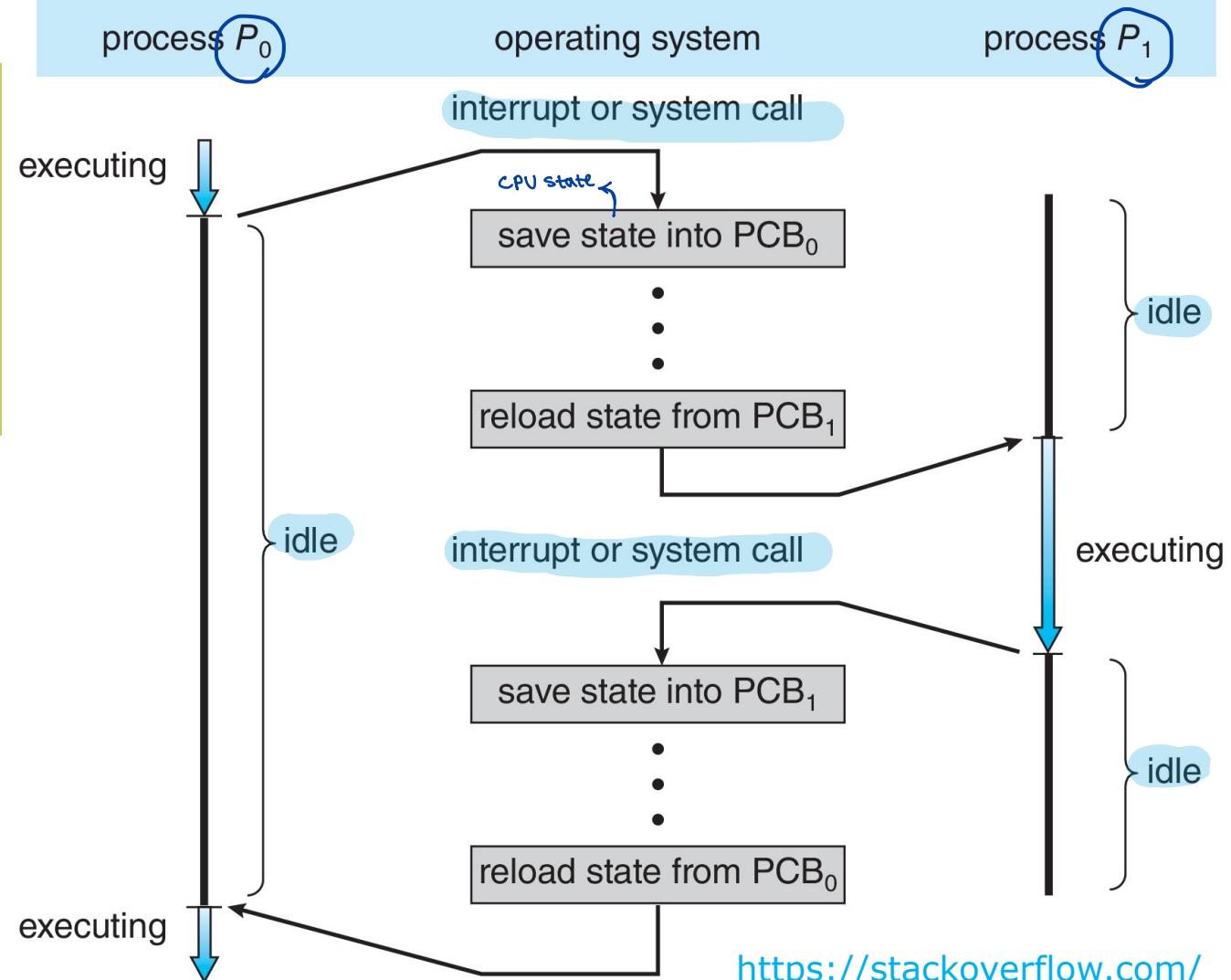
Seyyed Ahmad Javadi

[sajavadi@aut.ac.ir](mailto:sajavadi@aut.ac.ir)

Fall 2021

# CPU Switch From Process to Process

A **context switch** occurs when the CPU switches from one process to another.

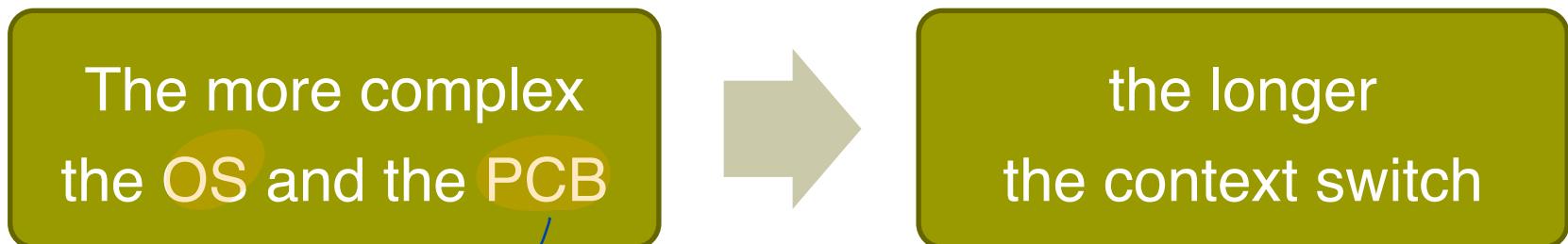


[https://stackoverflow.com/  
questions/9238326/system  
-call-and-context-switch](https://stackoverflow.com/questions/9238326/system-call-and-context-switch)



# Context Switch

- The system must *save the state* of the old process and load the *saved state* for the new process via a *context switch*.
- Context* of a process represented in the *PCB*.  
*process control block*
- Context-switch time is *pure overhead* →  
زبانی نهاد را که مکانیزم تغییر حالت پردازش را در میان پردازنده های مختلف ایجاد می کند، میتوان با عنوان *context switch* نامید.
- The system does no useful work while switching.



load > store زمان را زیاد نمایند  
هر دفعه PCB را بروزرسانی کنند  
و نمایند



# Context Switch (cont.)

## ■ Time dependent on hardware support

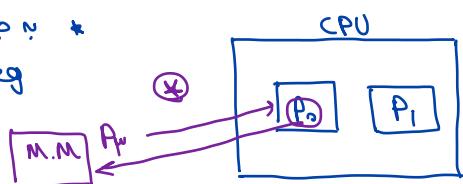
سی راه حل هستند؟  
 چهی وغیره هایی در میانشان!



Some hardware provides multiple sets of registers per CPU

multiple contexts loaded at once

\* با جای اینجا نیز چیزی که reg هایی از reg هایی باشند، چند reg هایی باز همان reg هایی باشند.



در هالت حادی برای جایگزینی بین  $P_1, P_2 \dots P_n$  را باید  $P_i$  State و  $M.M$  را با  $P_j$  State و  $M.M$  را باید آنچنانکه  $P_i$  State و  $M.M$  را باید آنچنانکه  $P_j$  State و  $M.M$  را باید تغییر دانیم. این خوندن را state در  $m.m.m$  می‌نامیم. این مبنای رسانیده است از reg هایی که باز همان reg هایی باشند، نویکی از یقینها  $P_{CB}$  و نویکی  $P_{CB}$  روز چندها کلمم، کامپیوئن رفت ترمه. دلیل طلاطای خوندن منزه شدن صرف نشد و  $P_1, P_2, P_3, P_4, P_5, P_6, P_7$  هاسته باشند، تک از  $P_1, P_2, P_3, P_4, P_5, P_6, P_7$  مثلاً خواسته شده است  $P_1$  را در آن  $m.m.m$  خوندن را با  $P_2$  را بخواست  $M.M$  از  $m.m.m$  خوندن را با  $P_3$  را بخواست دلیل بارم بنتاز خالت جبلی.



# Operations on Processes

- System must provide mechanisms for:
  - Process creation
  - Process termination

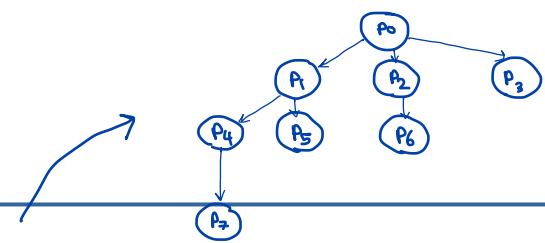
PCB با هم چه کاری نیستند؟ خیر

(IO, reg, ...)  
اون اطلاعات سربربط با مریدان هست.

(heap, stack, ...)  
اون مقداری مفهومی با هر بروزرسان هست.

→ address space \*

# Process Creation



- Parent process create children processes, which, in turn create other processes, forming a tree of processes.

- Process identified and managed via a process identifier (pid).

☞ هر پردازه یک سنتاپسی یعنی نام PID دارد.

- Resource sharing options

\* پردازه ها آدرس اسپیس های خالی کلمگ جز از نهم دارند. اون *thread* آدرس اسپیس، *pcb* رجیستر و استرک میزبانند.  
\* آن با استفاده از ترمینال یا برنامه ریجیستر می شوند.  
\* *terminal* = پردازه برنامه = نرمال.  
\* باستفاده از ترمینال، برنامه های کم با استفاده از ترمینال بازگردانی شده بخواهند.

- option 1 • Parent and children share all resources

- option 2 • Children share subset of parent's resources

- option 3 • Parent and child share no resources

- Execution options

سینی به معنای همانچه هر دو اجرا شوند.

هم زند

- option 1 • Parent and children execute concurrently

./office.exe &

\* آن خود از command بالا نه یعنی از ترمینال مخدوم اف داشته باشد معمول مسئله ای نباشد.

\* option 2 نیز ممکن است

☞ هر موقع اجرای کامپیوچر نئے نیز معمول مسئله ای نباشد.

# Process Creation (Cont.)

## ■ Address space

حالات ۱ و ۲ میان child و parent در فضای آدرسی:

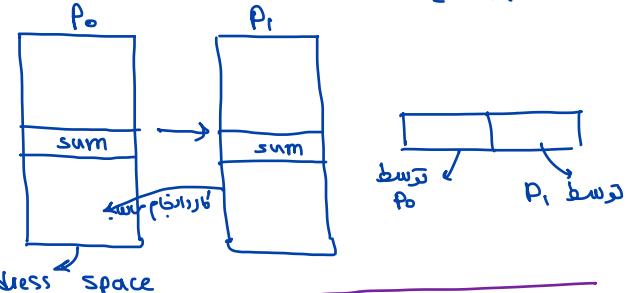
- Child **duplicate of parent**
- Child has a program loaded into it

parent با child در فضای آدرسی:

## ■ UNIX examples

- **fork()** system call creates new process.
- **exec()** system call used after a **fork()** to replace the process' memory space with a new program.
- Parent process calls **wait()** waiting for the child to terminate.

کم از انفع fork برای ایجاد کننده برخا را می خواهد. copy از مؤلفه های مسازه است.



وقت تقریبی اولین بروزگردانی را در اینجا لذت ببرید. اینجا P0 با P1 در فضای آدرسی متفاوت نیستند.

address space

با مداری (fork) این دو زندگی ارث می بردند. وقتی process id مطابق شود، exec (program) تغییر می کند.

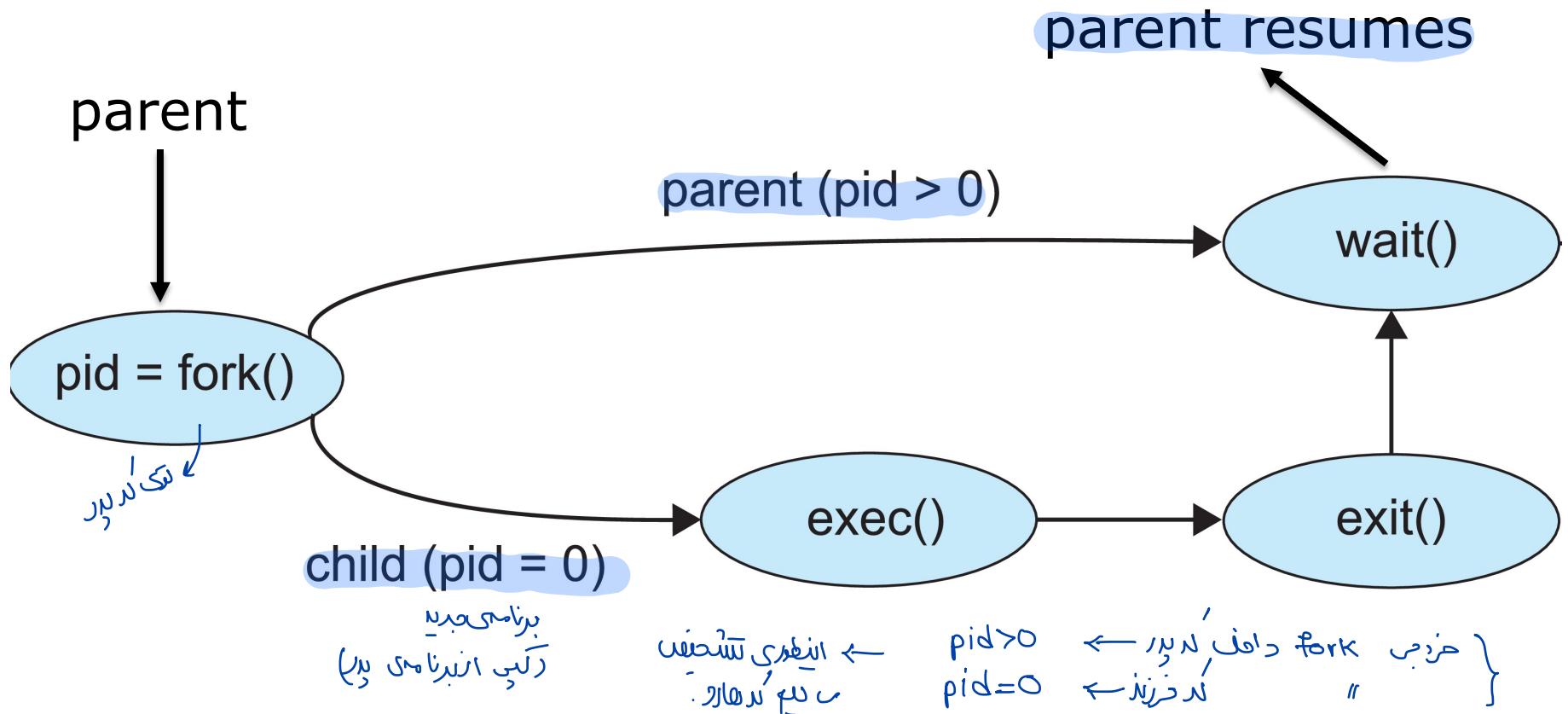
• exec() system call used after a fork() to replace the process'

memory space with a new program.

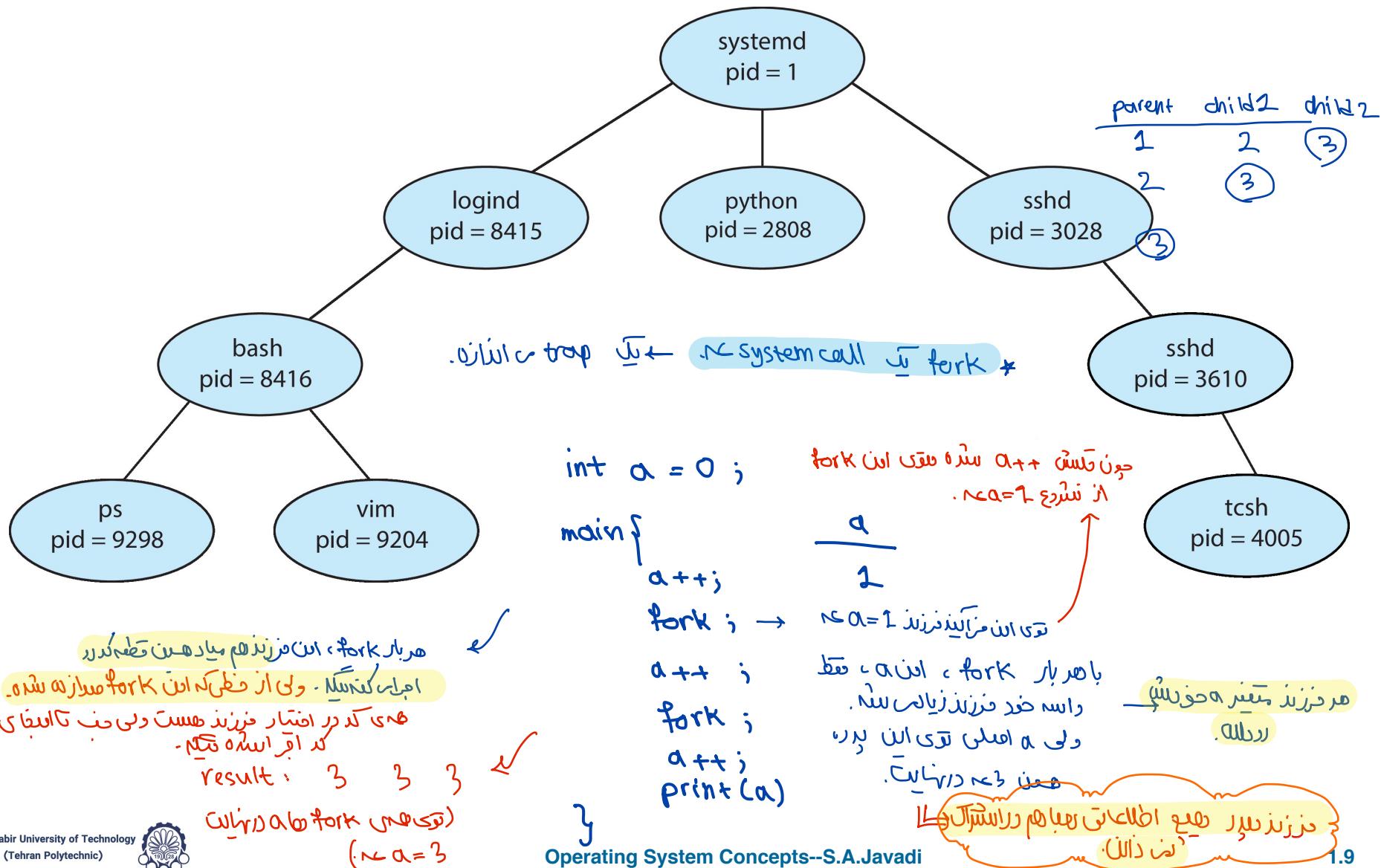
• Parent process calls wait() waiting for the child to terminate.



# Process Creation (Cont.)



# A Tree of Processes in Linux



# C Program Forking Separate Process

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

quit ← توي ← داخف  
for دخود for سرط

fork بذارم .  
لچور اللو هرین کنید!

لدنزند زاینها سورج با اجراس است

آن ایفا اینجا بذارم ، حکماً قتل از

child complete

wait



بیونت اول این خوب رسکن .

```
int main()  
{  
    pid_t pid;
```

\* در این pid هفته فرزندی عد 0< میلی توي لدنزند  
\* pid مسنه تا شخیش بگویند لدنزند .

linux میلی systemcall

/\* fork a child process \*/

pid = fork(); → کم از پردازه ساخته می شود pid

```
if (pid < 0) { /* error occurred */  
    fprintf(stderr, "Fork Failed");  
    return 1;  
}
```

```
else if (pid == 0) { /* child process */  
    execlp("/bin/ls", "ls", NULL);  
    printf("child");
```

```
else { /* parent process */  
    /* parent will wait for the child to complete */
```

wait(NULL);

printf("Child Complete");

child توی بینامی fork  
صفیدم گردید ، این  
نهست نه توی child  
اجراس نم

```
return 0;  
}
```

این فهمت  
در پردازند پر اجرام می شوند .  
منتظر عده می شوند پر می شوند .  
خون توي لور > pid مسنه .



# Process Termination

---

- Process executes last statement and then asks the operating system to delete it using the `exit()` system call.
  - Returns status data from child to parent (via `wait()`)
  - Process' resources are deallocated by operating system.
- Parent may terminate the execution of children processes using the `abort()` system call. Some reasons for doing so:
  - Child has exceeded allocated resources.
  - Task assigned to child is no longer required.
  - The parent is exiting, and the operating systems does not allow a child to continue if its parent terminates.

# Process Termination (Cont.)

---

- Some OSs do not allow child to *exists* if its parent has terminated.
  - If a process terminates, then all its children must also be terminated.
  - **Cascading termination:** All children, grandchildren, etc., are terminated.
  - The termination is initiated by the operating system.



# Process Termination (Cont.)

---

- The parent process may wait for termination of a child process by using the ***wait() system call.***
  - The call returns status information and the pid of the terminated process.

***pid = wait(&status);***

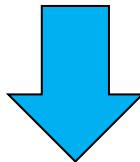
- If no parent waiting (did not invoke wait()) process is a **zombie**.
- If parent terminated without invoking wait(), process is an **orphan**.

## Multiprocess Architecture – Browser (Cont.)

---

- Many web browsers ran as single process (some still do)

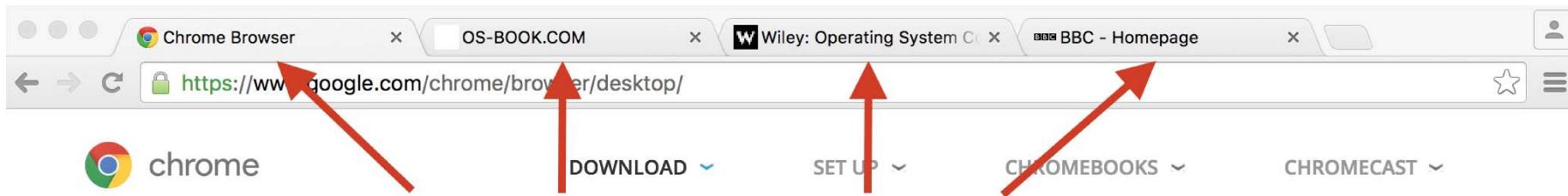
If one web site causes trouble



Entire browser can hang or crash

# Multiprocess Architecture – Chrome Browser (Cont.)

- Google Chrome is multiprocess with 3 different types of processes:
  - **Browser** process manages user interface, disk and network I/O.
  - **Renderer** process renders web pages, deals with HTML, Javascript.
    - ▶ A new renderer created for each website opened
    - ▶ Runs in **sandbox** restricting disk and network I/O, minimizing effect of security exploits.
  - **Plug-in** process for each type of plug-in.



Each tab represents a separate process.