



One CPU per control flow

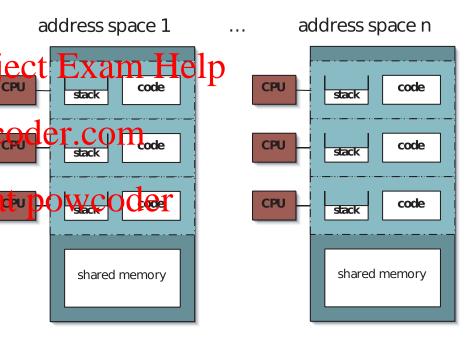
Specialized configurations e.g.
 Distributed properties

Distributed properties

– Physical process control systems: 1 CPU per task, connected vil QW COLOR COM bus-system

 Process management dd WeChattopowc (scheduling) not required

 Must coordinate access to shared memory

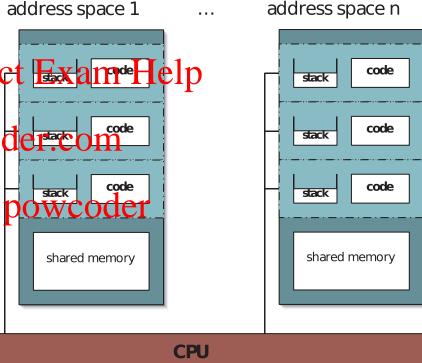


One CPU for all control flows

OS: emulate one CPU for every control flow: Assignment Project ExameH

Multi-tasking operating system stack https://powcoder.com*

- Process management (scheduling) required dd WeChat
- Must coordinate access to shared memory
- Support for memory protection essential



Processes address space 1 address space n Process ::= address space + control flow(s) Assignment Project Exam Help code stack • Kernel has full knowledge https://powcoder.com code stack about all processes ntt code code state stack Add WeChat powco requirements - currently held resources shared memory shared memory **CPU**

User-level scheduling

Parent process performs I/O

Processes and Threads C03

shared memory

Threads address space 1 address space n • Threads (individual control flows) can be handled ment Project Exam Help thread stack – Inside the OS: • Kernel scheduling ttps://powcoder. thread stack • Thread performs / O WeChat powc thread stack - Outside the OS:

shared memory

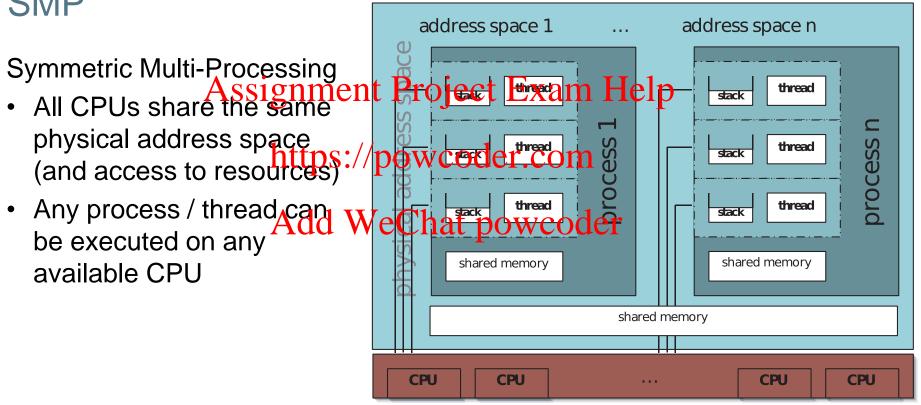
CPU



SMP

Symmetric Multi-Processing

- physical address space (and access to resources)
- Any process / thread can be executed on any available CPU





Processes or Threads?

Specific definition of threads depends on operating system and context:
 Threads ≡ a group of processes, which share some resources

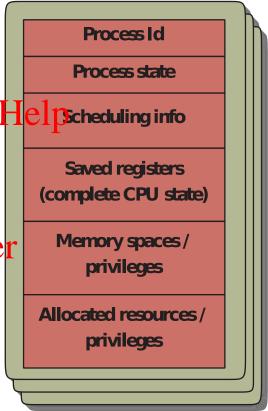
- (process hierarchy) https://powcoder.com
 Due to the overlap in resources, the attributes attached to threads are less than for 'first-class-citizen-processes'

 Thread switching and inter-thread communication can be more
- efficient than switching on process level
- Scheduling of threads depends on the actual thread implementations:
 - user-level control-flows (no kernel support)
 - kernel-level control-flows, handled as processes with some restrictions



Process Control Blocks

- Process Id
- Process state: Assignment Project Exam Helpscheduling info {created, ready, executing, blocked, https://powcoder.com suspended, ...}
- Scheduling attributes: Add WeChat powcoder Priorities, deadlines, consumed CPU-time, ...
- CPU state: Saved/restored information on context switches (including program counter, stack pointer, ...)



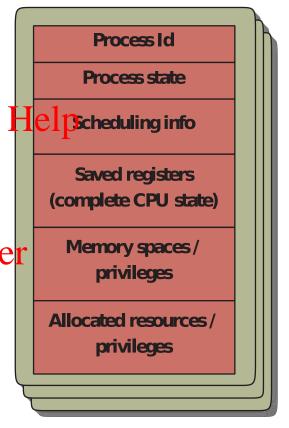


Process Control Blocks

• Memory attributes / privileges: Project Exam Helpscheduling info memory base, limits, shared areas, ...der.com

 Allocated resources / privileges: open and requested devices and files.

PCBs are commonly enqueued at a certain state or condition (awaiting access or change in state)





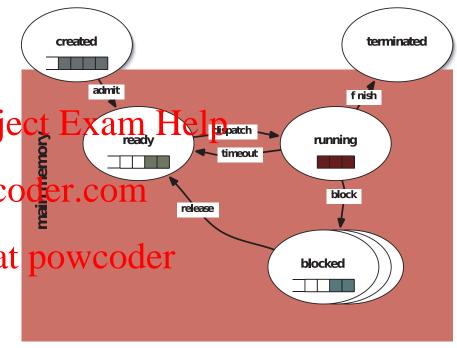
Process States

task is ready to the sufficient Project Exam considered by any dispatcher (waiting for admission) tps://powcoder.com

• ready: ready to run (waiting fold WeChat powcoder

 running: holding a CPU and executing

 blocked: not ready to run (waiting for a resource)

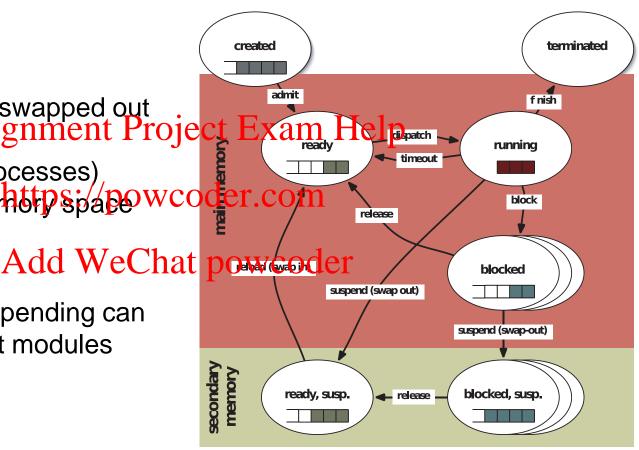




Process States

suspended states: swapped out of main memory ssignment Project Example of main memory suspended out of (non-time-critical processes) waiting for main merhot psparewcoder.com (or other resources)

 dispatching and suspending can now be independent modules





Process States

pre-emption or cycle done executing batch __ready Exinam SS1 239 Title creation termination https://eag.w.speder.com suspend (swap-out) Add We Chat powcoder blocked, suspended swap-out blocked block or synchronize



Unix Processes

• In UNIX systems tasks are created by 'cloning'ignment Project Exam Help

```
parent child

fork

wait

exec
```

```
pid = fork();
```

- duplicates the current process powcoder comila process
- returns 0 to the newly created process (the 'child' process Chat powcoder
- returns the process id of child process to creating process (the 'parent' process)
- or returns -1 as C-style indication of failure

```
exec("path_to_exe", args);
exit(0); // terminate

OVCOCET

// parent process
...
pid = wait();
/* wait for termination of one child process */
}
```



Communication Between Unix Processes

```
else { // parent/sender
int pipe fd[2], c, rc;
if (pipe (pipe_fd) Assignment Projectifix (ain Field);

(c) = getchar()) > 0) {
  perror("no pipe"); exit(1);
                                              if (write(pipe fd[1], &c, 1) == -1)
if (fork() == 0) { // childreceiper wcoder.com
perror("pipe broken");
                                              close(pipe fd[1]); exit(1);
  close(pipe fd[1]);
  while ((rc = read (pipe_Add, WeChat powcoder
1)) > 0) putchar(c);
                                            close(pipe fd[1]);
  if (rc == -1) {
                                            wait(NULL);
                                                            parent
                                                                       pipe
                                                                               child
    perror("pipe broken");
                                                                  pipe
    close(pipe fd[0]); exit(1);
                                                                           fork
                                                                 write
                                                                           read
  close(pipe fd[0]);
                                                                 close
                                                                           close
```