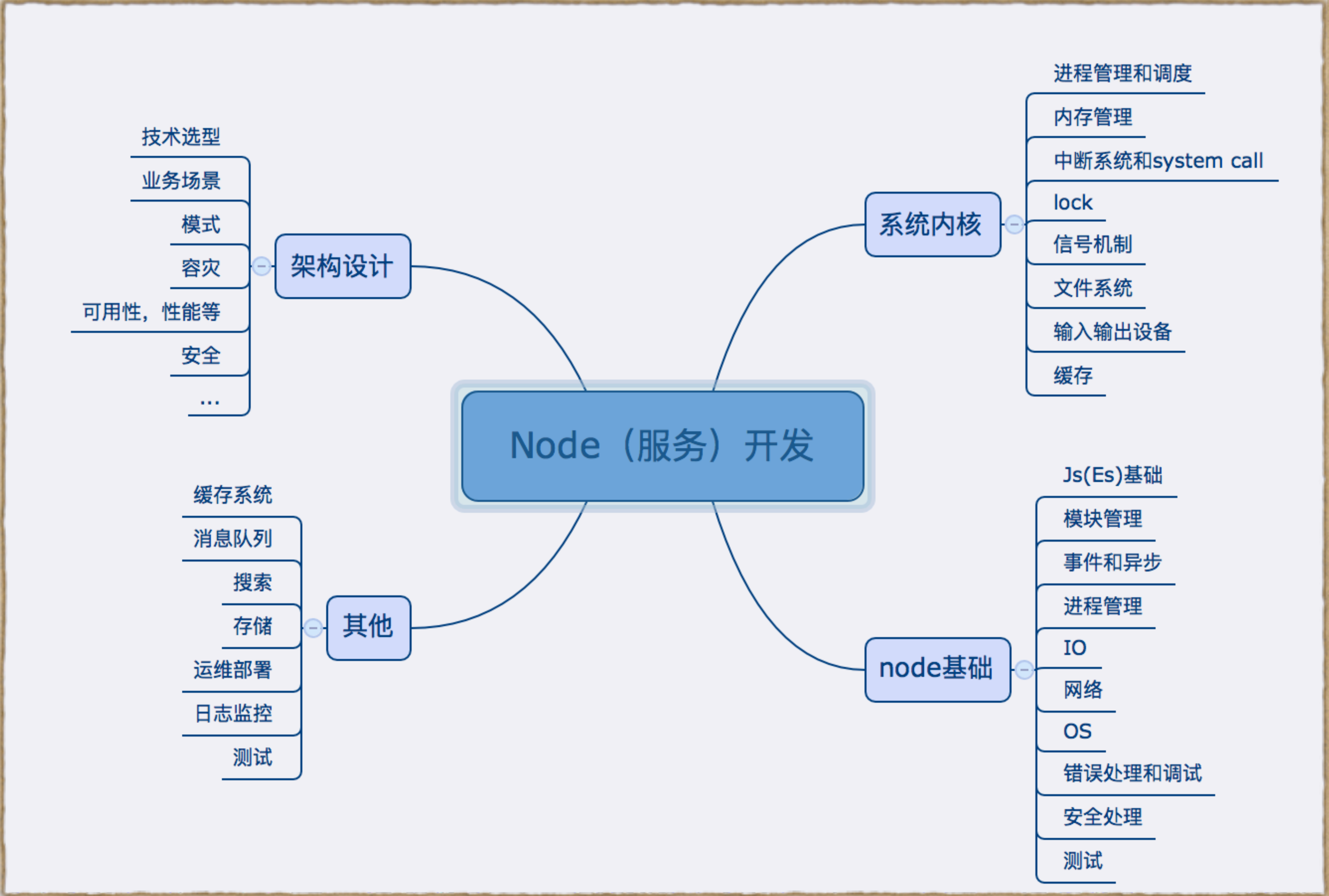
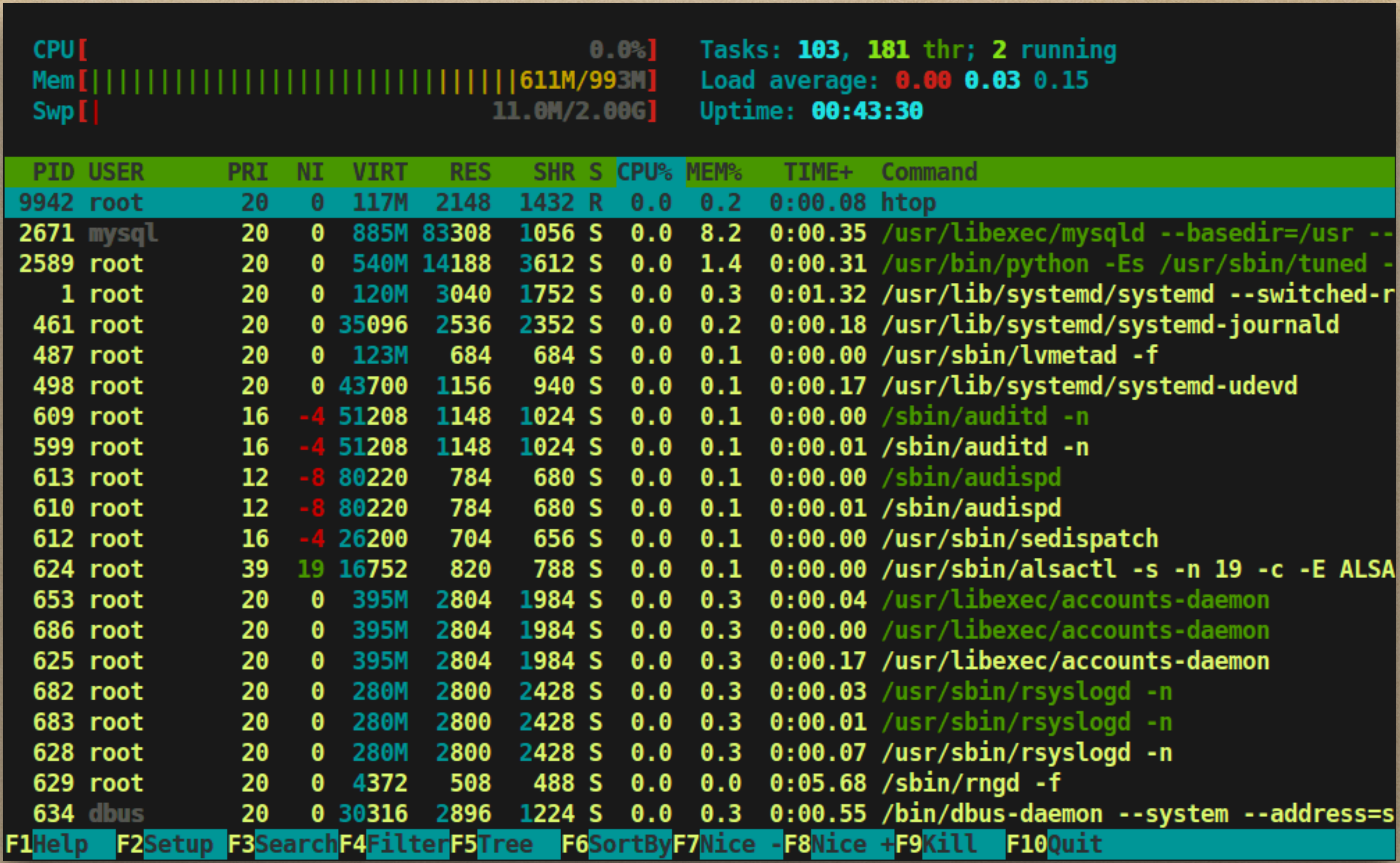
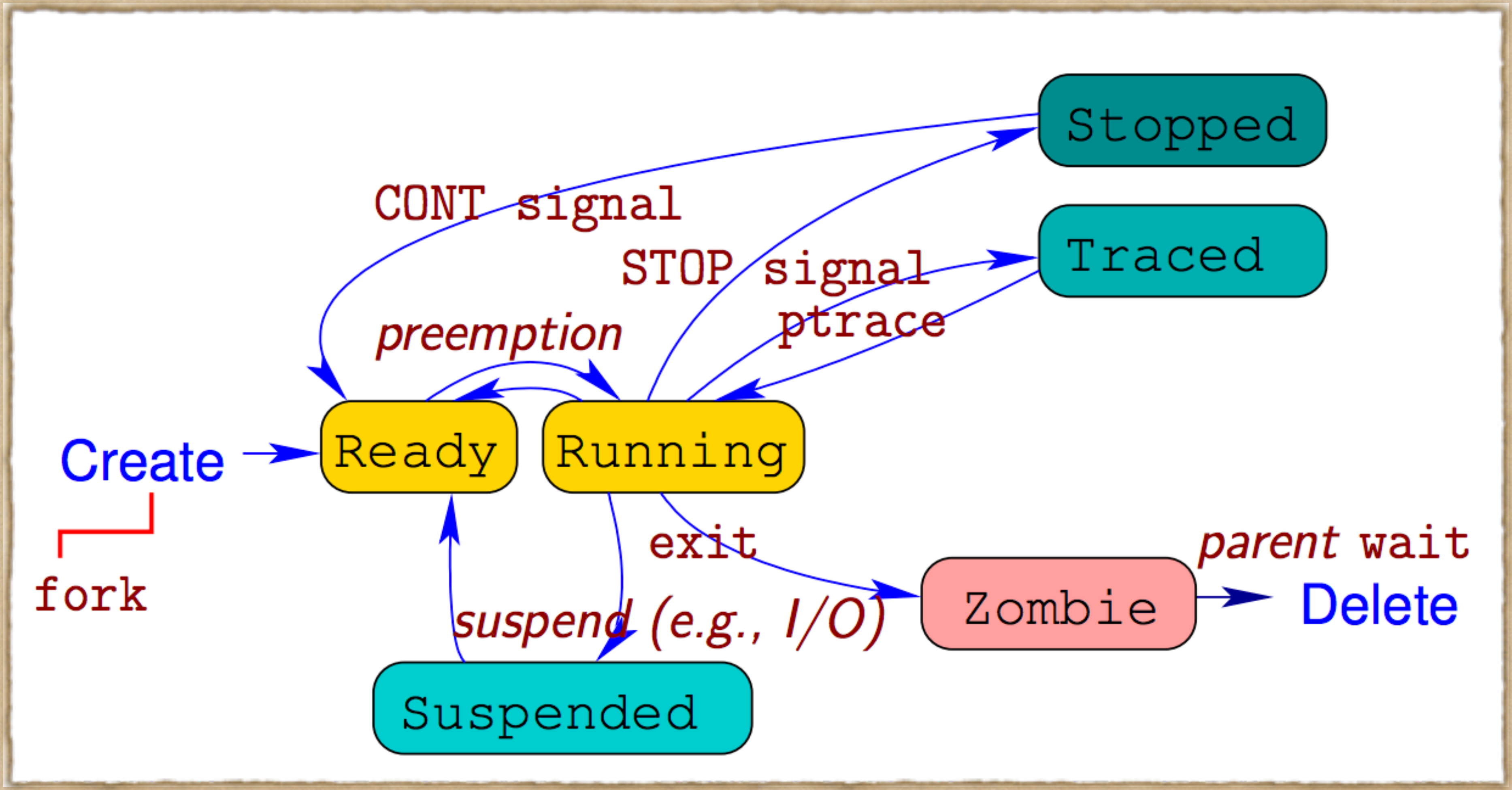
Node 进程管理


Node（服务）开发
架构设计
技术选型
业务场景
模式
容灾
可用性，性能等
安全
...
系统内核
进程管理和调度
内存管理
中断系统和system call
lock
信号机制
文件系统
输入输出设备
缓存
其他
缓存系统
消息队列
搜索
存储
运维部署
日志监控
测试
node基础
Js(Es)基础
模块管理
事件和异步
进程管理
IO
网络
OS
错误处理和调试
安全处理
测试



```
  CPU[                                     0.0%]   Tasks: 103, 181 thr; 2 running
  Mem[|||||||||||||||||||||||||||||||611M/993M]   Load average: 0.00 0.03 0.15
  Swp[|                              11.0M/2.00G]   Uptime: 00:43:30

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
 9942 root       20   0  117M  2148  1432 R  0.0  0.2  0:00.08 htop
 2671 mysql      20   0  885M 83308  1056 S  0.0  8.2  0:00.35 /usr/libexec/mysqld --basedir=/usr --
 2589 root       20   0  540M 14188  3612 S  0.0  1.4  0:00.31 /usr/bin/python -Es /usr/sbin/tuned -
    1 root       20   0  120M  3040  1752 S  0.0  0.3  0:01.32 /usr/lib/systemd/systemd --switched-r
  461 root       20   0 35096  2536  2352 S  0.0  0.2  0:00.18 /usr/lib/systemd/systemd-journald
  487 root       20   0  123M   684   684 S  0.0  0.1  0:00.00 /usr/sbin/lvmetad -f
  498 root       20   0 43700  1156   940 S  0.0  0.1  0:00.17 /usr/lib/systemd/systemd-udevd
  609 root       16  -4 51208  1148  1024 S  0.0  0.1  0:00.00 /sbin/auditd -n
  599 root       16  -4 51208  1148  1024 S  0.0  0.1  0:00.01 /sbin/auditd -n
  613 root       12  -8 80220   784   680 S  0.0  0.1  0:00.00 /sbin/audispd
  610 root       12  -8 80220   784   680 S  0.0  0.1  0:00.01 /sbin/audispd
  612 root       16  -4 26200   704   656 S  0.0  0.1  0:00.00 /usr/sbin/sedispatch
  624 root       39  19 16752   820   788 S  0.0  0.1  0:00.00 /usr/sbin/alsactl -s -n 19 -c -E ALSA
  653 root       20   0  395M  2804  1984 S  0.0  0.3  0:00.04 /usr/libexec/accounts-daemon
  686 root       20   0  395M  2804  1984 S  0.0  0.3  0:00.00 /usr/libexec/accounts-daemon
  625 root       20   0  395M  2804  1984 S  0.0  0.3  0:00.17 /usr/libexec/accounts-daemon
  682 root       20   0  280M  2800  2428 S  0.0  0.3  0:00.03 /usr/sbin/rsyslogd -n
  683 root       20   0  280M  2800  2428 S  0.0  0.3  0:00.01 /usr/sbin/rsyslogd -n
  628 root       20   0  280M  2800  2428 S  0.0  0.3  0:00.07 /usr/sbin/rsyslogd -n
  629 root       20   0  4372   508   488 S  0.0  0.0  0:05.68 /sbin/rngd -f
  634 dbus       20   0 30316  2896  1224 S  0.0  0.3  0:00.55 /bin/dbus-daemon --system --address=s
F1Help  F2Setup F3SearchF4FilterF5Tree  F6SortByF7Nice -F8Nice +F9Kill  F10Quit
```

什么是进程?

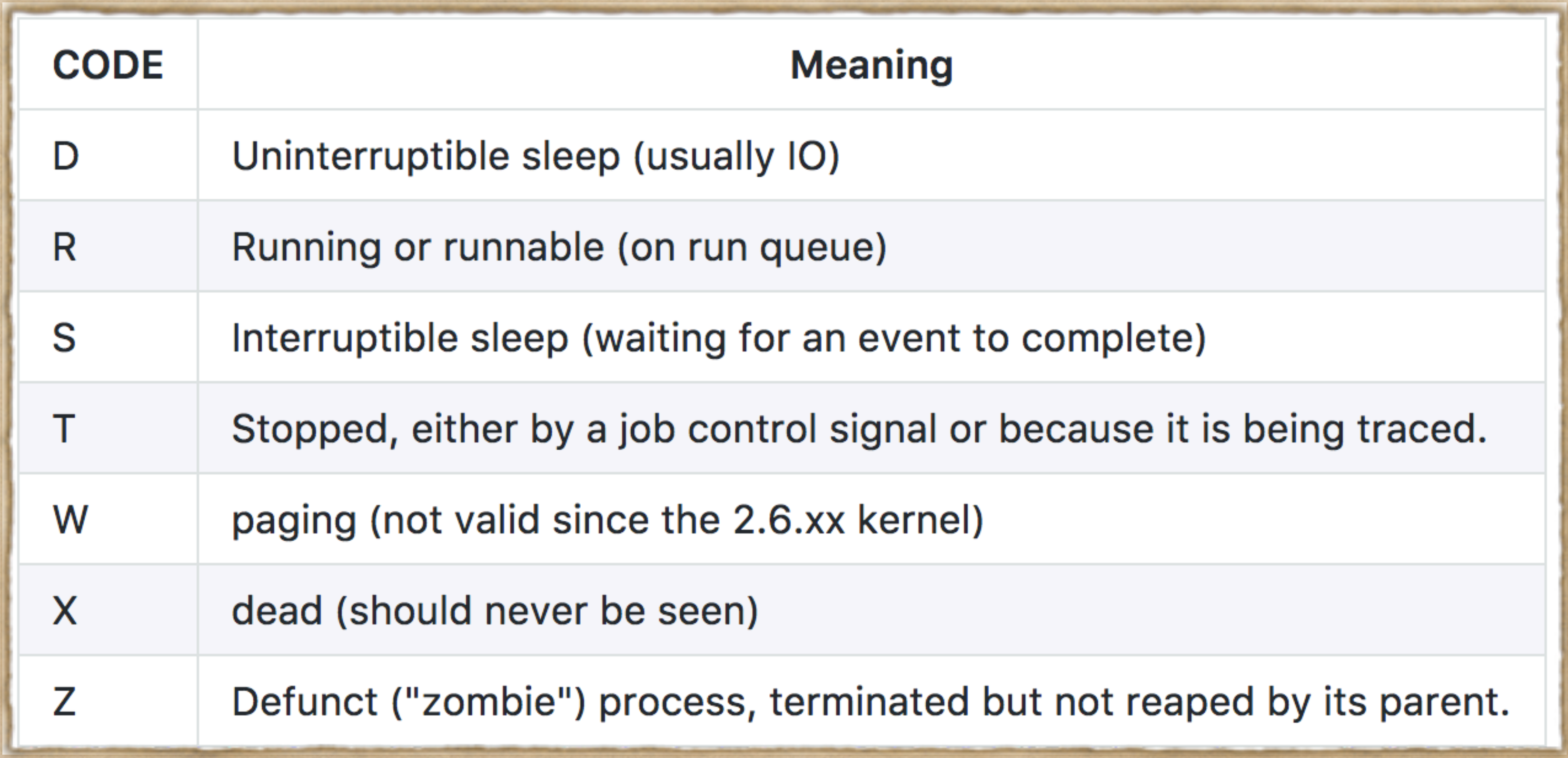


a process is an instance of a computer program that is being executed

进程查看

- ps -ef | grep node (ps -aux)
- top

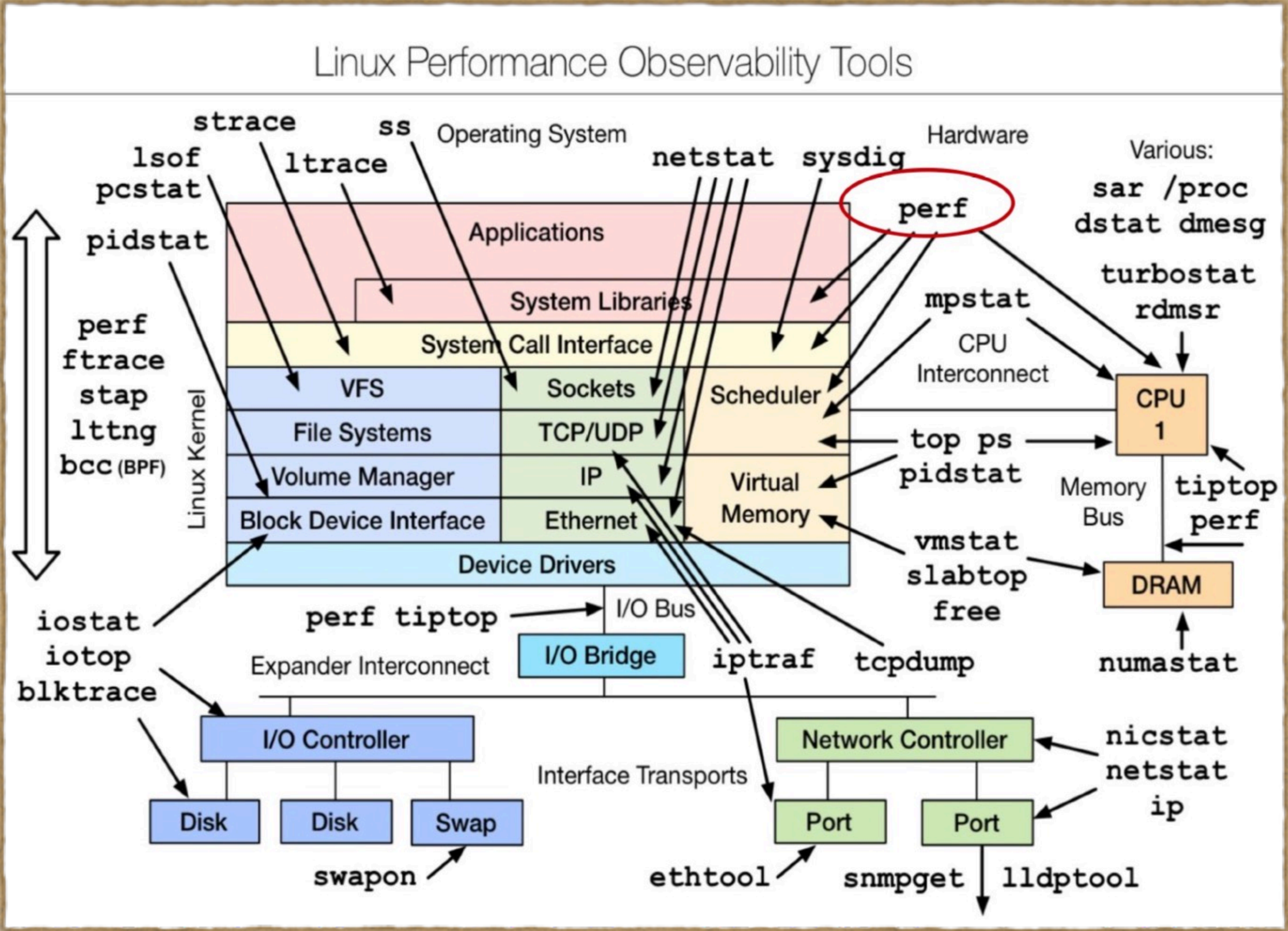
状态码



CODE	Meaning
D	Uninterruptible sleep (usually IO)
R	Running or runnable (on run queue)
S	Interruptible sleep (waiting for an event to complete)
T	Stopped, either by a job control signal or because it is being traced.
W	paging (not valid since the 2.6.xx kernel)
X	dead (should never be seen)
Z	Defunct ("zombie") process, terminated but not reaped by its parent.

其他概念

- 线程(thread)，协程(coroutine)？


Linux Performance Observability Tools
strace
ss
Operating System
Hardware
Various:
lsof
ltrace
netstat
sysdig
sar /proc
pcstat
perf
dstat dmesg
pidstat
Applications
turbostat
System Libraries
mpstat
rdmsr
perf
ftrace
stap
lttng
bcc (BPF)
System Call Interface
CPU Interconnect
Linux Kernel
VFS
Sockets
Scheduler
CPU 1
File Systems
TCP/UDP
top ps
Volume Manager
IP
pidstat
Virtual Memory
Memory Bus
tiptop
perf
Block Device Interface
Ethernet
vmstat
slabtop
free
Device Drivers
DRAM
iostat
iotop
blktrace
perf tiptop
I/O Bus
I/O Bridge
iptraf
tcpdump
numastat
Expander Interconnect
I/O Controller
Network Controller
nicstat
netstat
ip
Interface Transports
Disk
Disk
Swap
Port
Port
swapon
ethtool
snmpget
lldptool

常用shell命令

node进程模块

- Process：进程介绍
- child_process：子进程&IPC
- cluster：负载均衡实现

- 进程基础信息
- 进程 Usage
- 进程级事件
- 系统账户信息
- 环境变量
- 信号收发
- 三个标准流
- Node.js 依赖模块/版本信息
- 其他

child_process

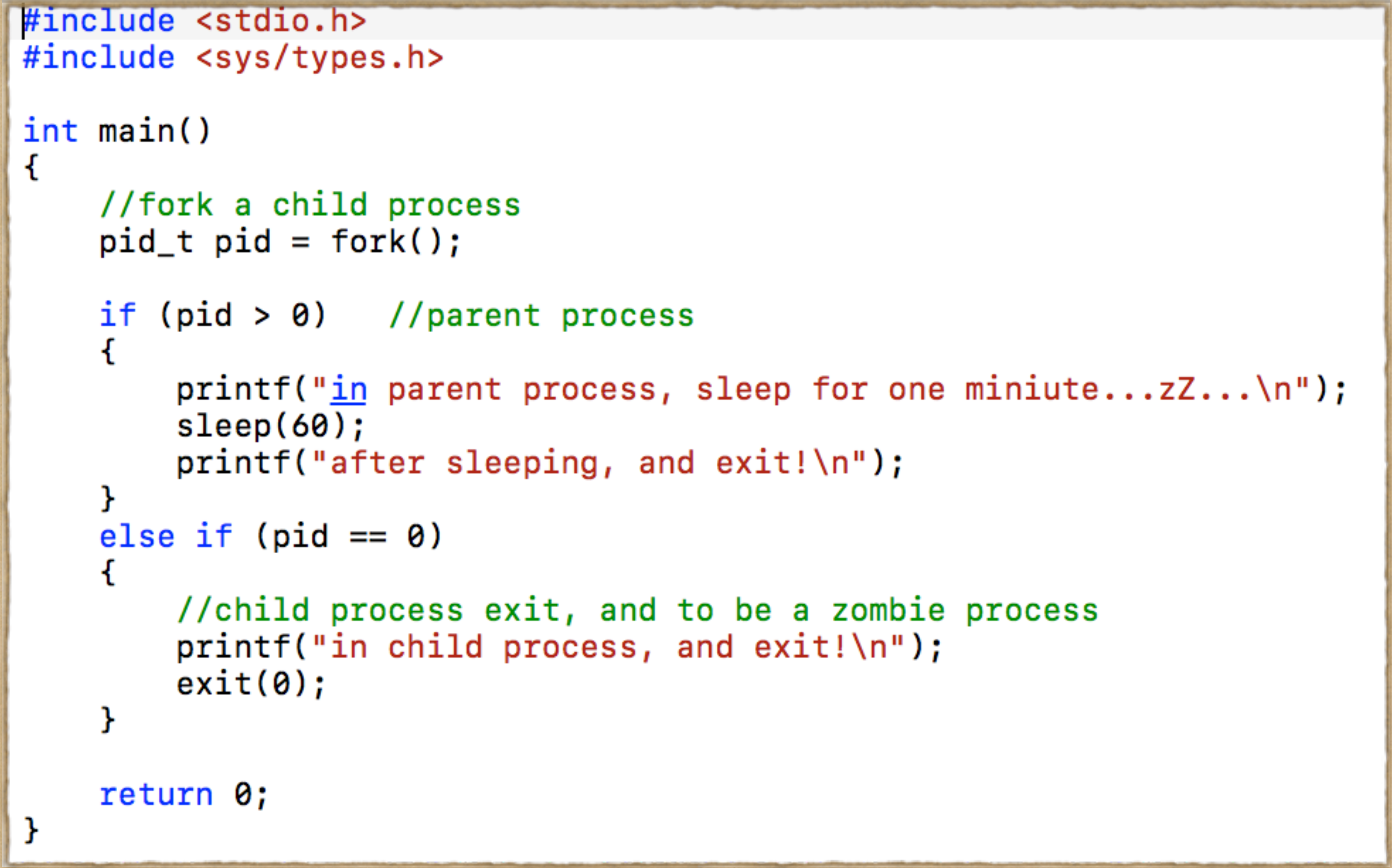
child_process方法

- `child_process.spawn(command, args)`
- `child_process.exec(command, options)`
- `child_process.execFile(file, args[, callback])`
- `child_process.fork(modulePath, args)`

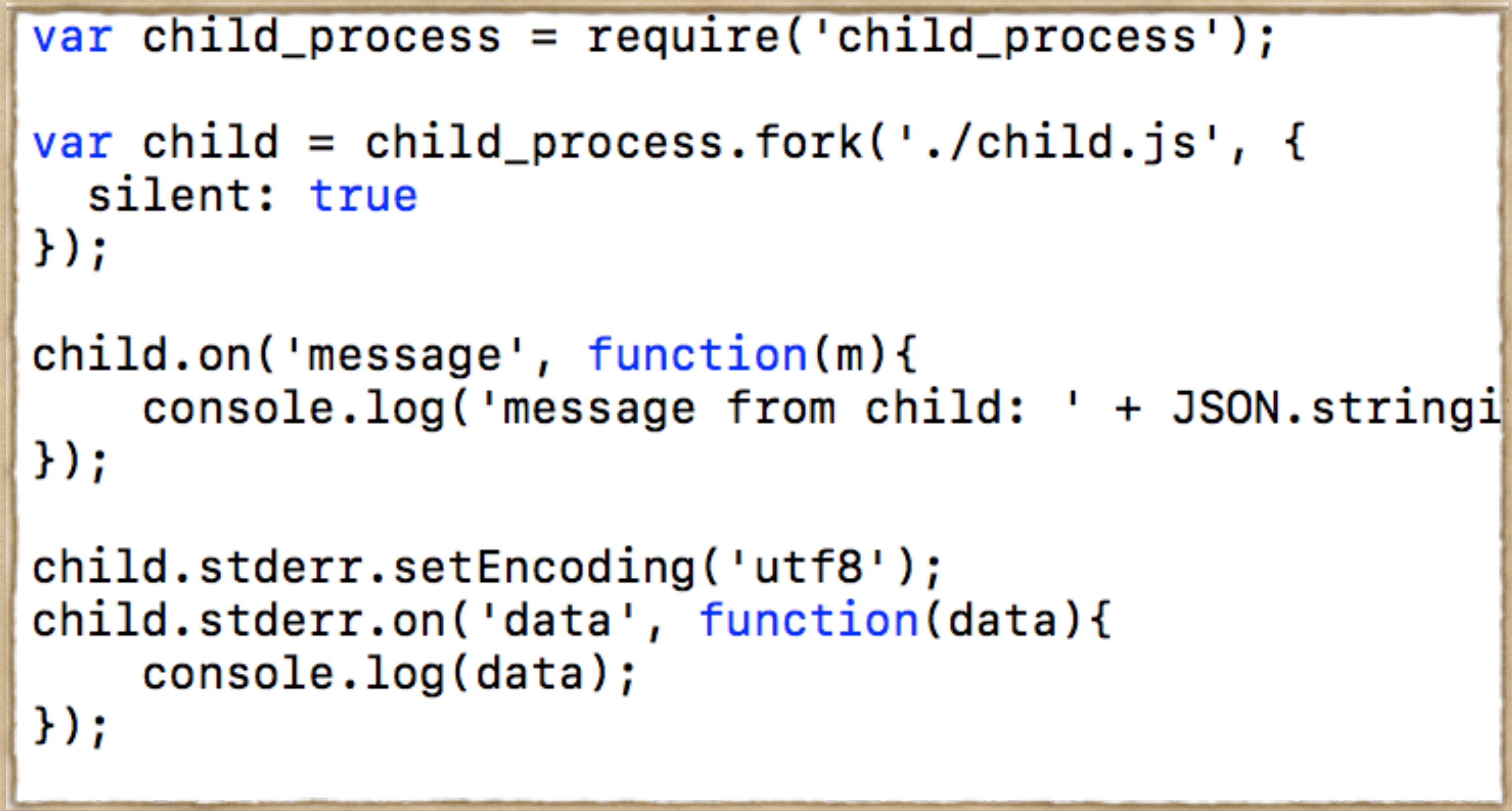
```javascript
const spawn = require('child_process').spawn;
const ls = spawn('ls', ['-l', '/']);

ls.stdout.on('data', (data) => {
  console.log(`stdout: ${data}`);
});
```

什么是子进程?

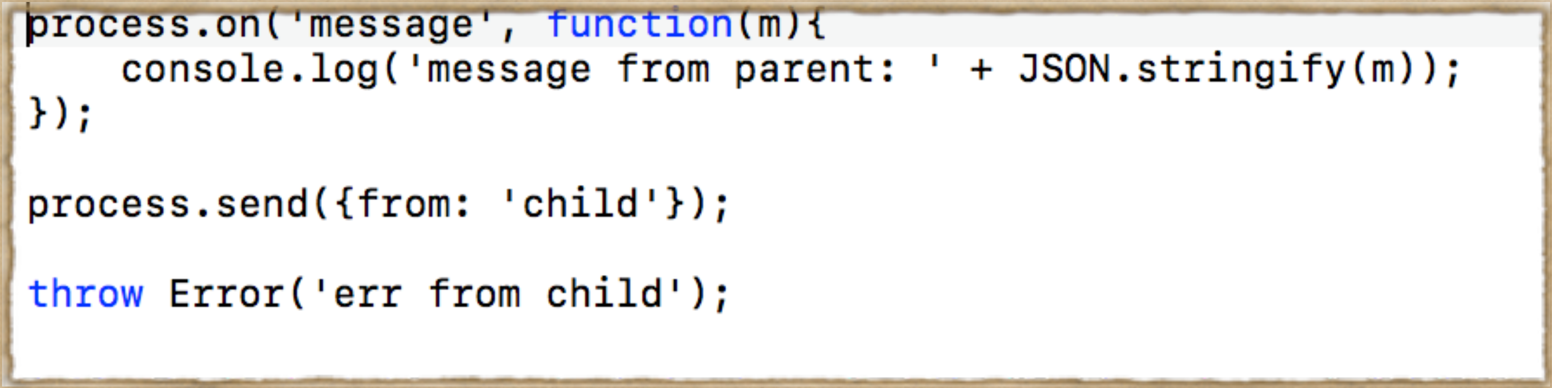
- 由另个线程(父进程)创建，复制父进程数据空间、堆和栈
- 一般存在于执行多任务的系统
- 创建方法：the fork system call (Unix-like systems and the POSIX standard) and the spawn ((NT) kernel of Microsoft Windows)



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

int main()
{
    //fork a child process
    pid_t pid = fork();

    if (pid > 0)   //parent process
    {
        printf("in parent process, sleep for one miniute...zZ...\n");
        sleep(60);
        printf("after sleeping, and exit!\n");
    }
    else if (pid == 0)
    {
        //child process exit, and to be a zombie process
        printf("in child process, and exit!\n");
        exit(0);
    }

    return 0;
}
```

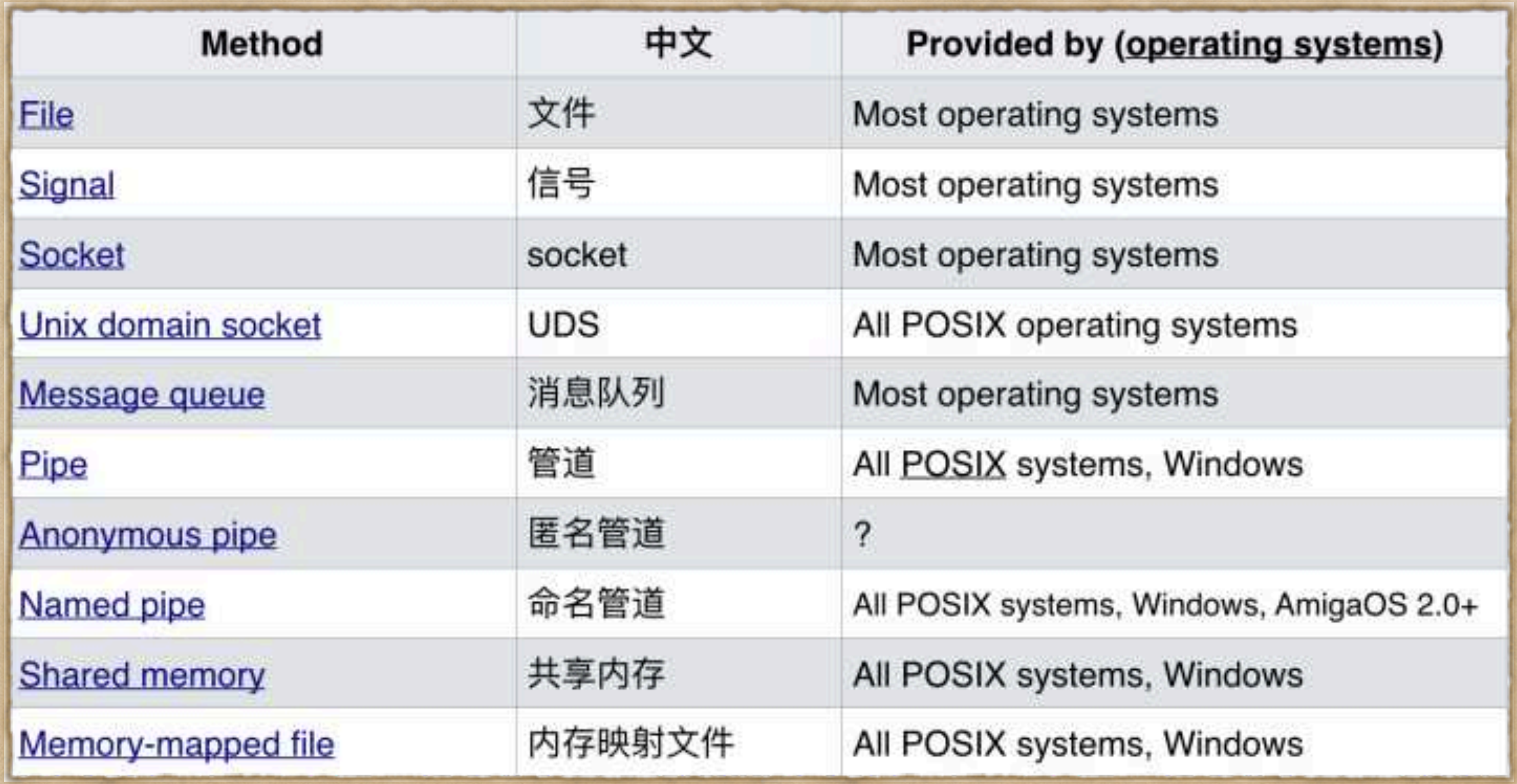


```
var child_process = require('child_process');

var child = child_process.fork('./child.js', {
  silent: true
});

child.on('message', function(m){
    console.log('message from child: ' + JSON.stringi
});

child.stderr.setEncoding('utf8');
child.stderr.on('data', function(data){
    console.log(data);
});
```



```
process.on('message', function(m){
    console.log('message from parent: ' + JSON.stringify(m));
});

process.send({from: 'child'});

throw Error('err from child');
```

通信方式

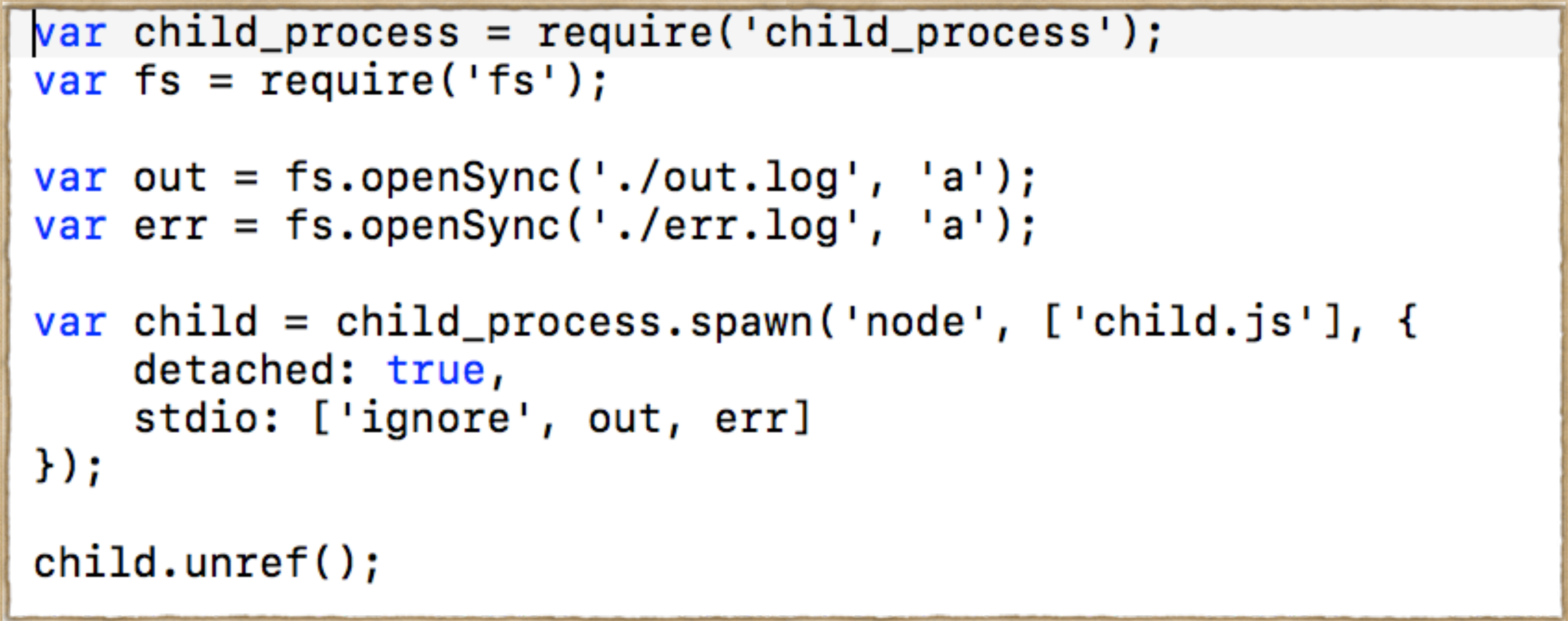


Method	中文	Provided by (operating systems)
File	文件	Most operating systems
Signal	信号	Most operating systems
Socket	socket	Most operating systems
Unix domain socket	UDS	All POSIX operating systems
Message queue	消息队列	Most operating systems
Pipe	管道	All POSIX systems, Windows
Anonymous pipe	匿名管道	?
Named pipe	命名管道	All POSIX systems, Windows, AmigaOS 2.0+
Shared memory	共享内存	All POSIX systems, Windows
Memory-mapped file	内存映射文件	All POSIX systems, Windows

stdio选项

```
'pipe' - equivalent to ['pipe', 'pipe', 'pipe'] (the default)

'ignore' - equivalent to ['ignore', 'ignore', ‘ignore']

'inherit' - equivalent to [process.stdin, process.stdout,
 process.stderr] or [0,1,2]
```

stdio重定向



```
var child_process = require('child_process');
var fs = require('fs');

var out = fs.openSync('./out.log', 'a');
var err = fs.openSync('./err.log', 'a');

var child = child_process.spawn('node', ['child.js'], {
    detached: true,
    stdio: ['ignore', out, err]
});

child.unref();
```

node Cluster

cluster多进程

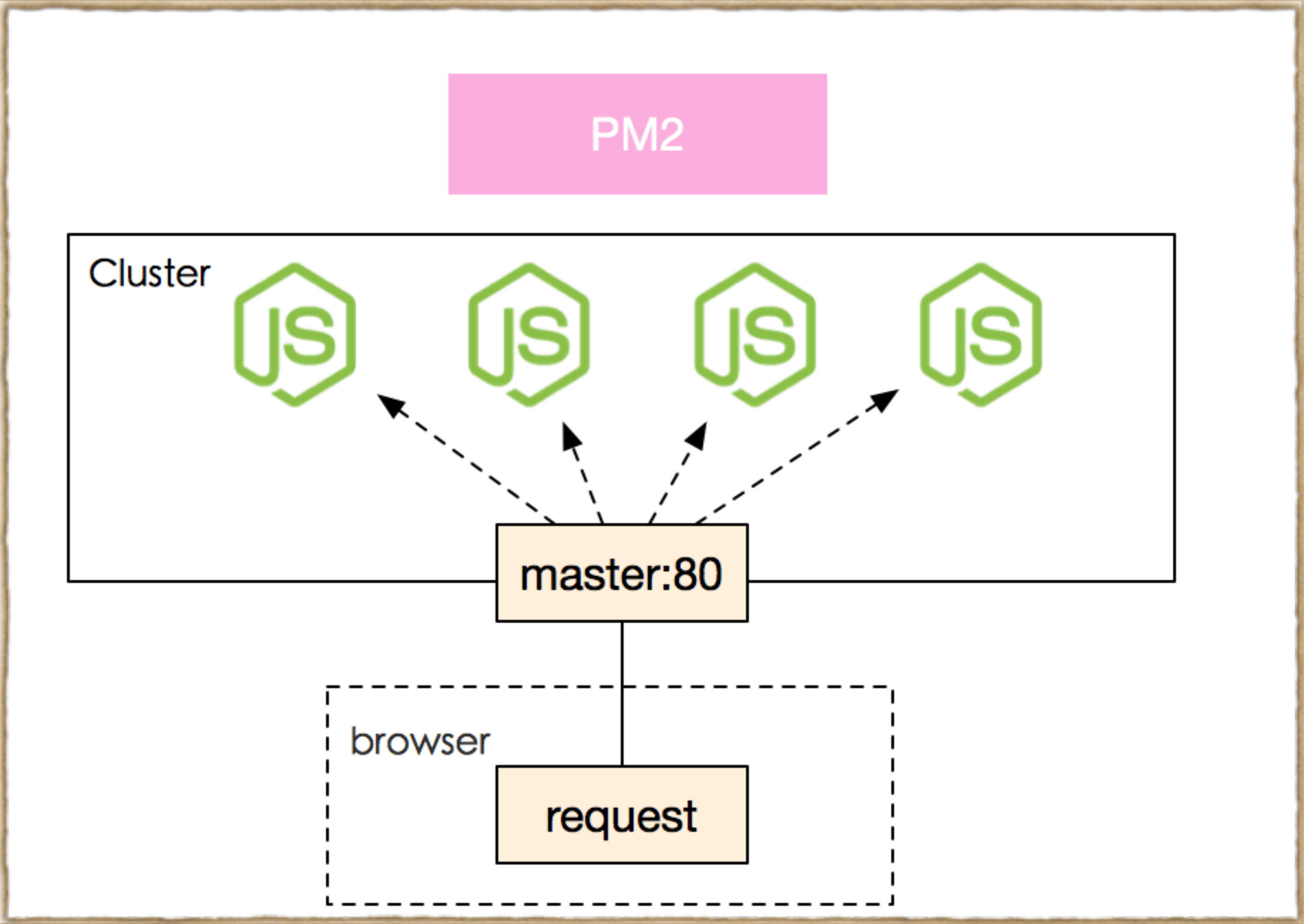
- 利用多核处理任务
- child_process.fork()实现
- cluster 产生的进程之间是通过 IPC 来通信

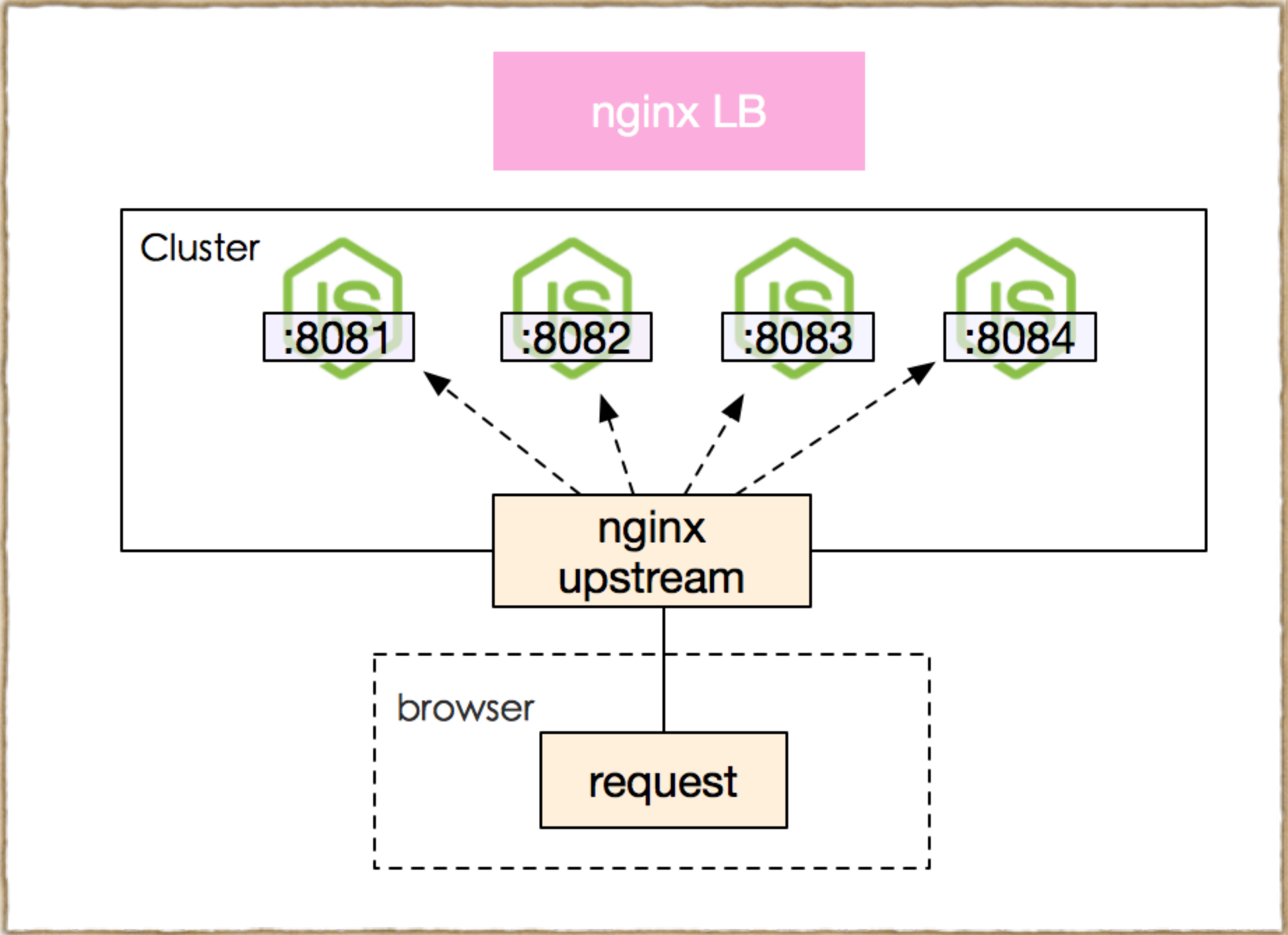
```
const cluster = require('cluster');            //  | |
const http = require('http');                  //  | |
const numCPUs = require('os').cpus().length;   //  | |    都执行了
                                               //  | |
if (cluster.isMaster) {                        //  |-|-----------------
  // Fork workers.                             //    |
  for (var i = 0; i < numCPUs; i++) {          //    |
    cluster.fork();                            //    |
  }                                            //    |  仅父进程执行
  cluster.on('exit', (worker) => {             //    |
    console.log(`${worker.process.pid} died`); //    |
  });                                          //    |
} else {                                       //  |-------------------
  // Workers can share any TCP connection      //  |
  // In this case it is an HTTP server         //  |
  http.createServer((req, res) => {            //  |

    console.log('imcoming req')
    res.writeHead(200);                        //  |    仅子进程执行
    res.end('hello world' + process.pid +'\n');                  // |
  }).listen(8000);                             //  |
}                                              //  |-------------------
                                               //  | |
console.log('hello '+process.pid);
```

LB算法

1.时间片轮转调度算法

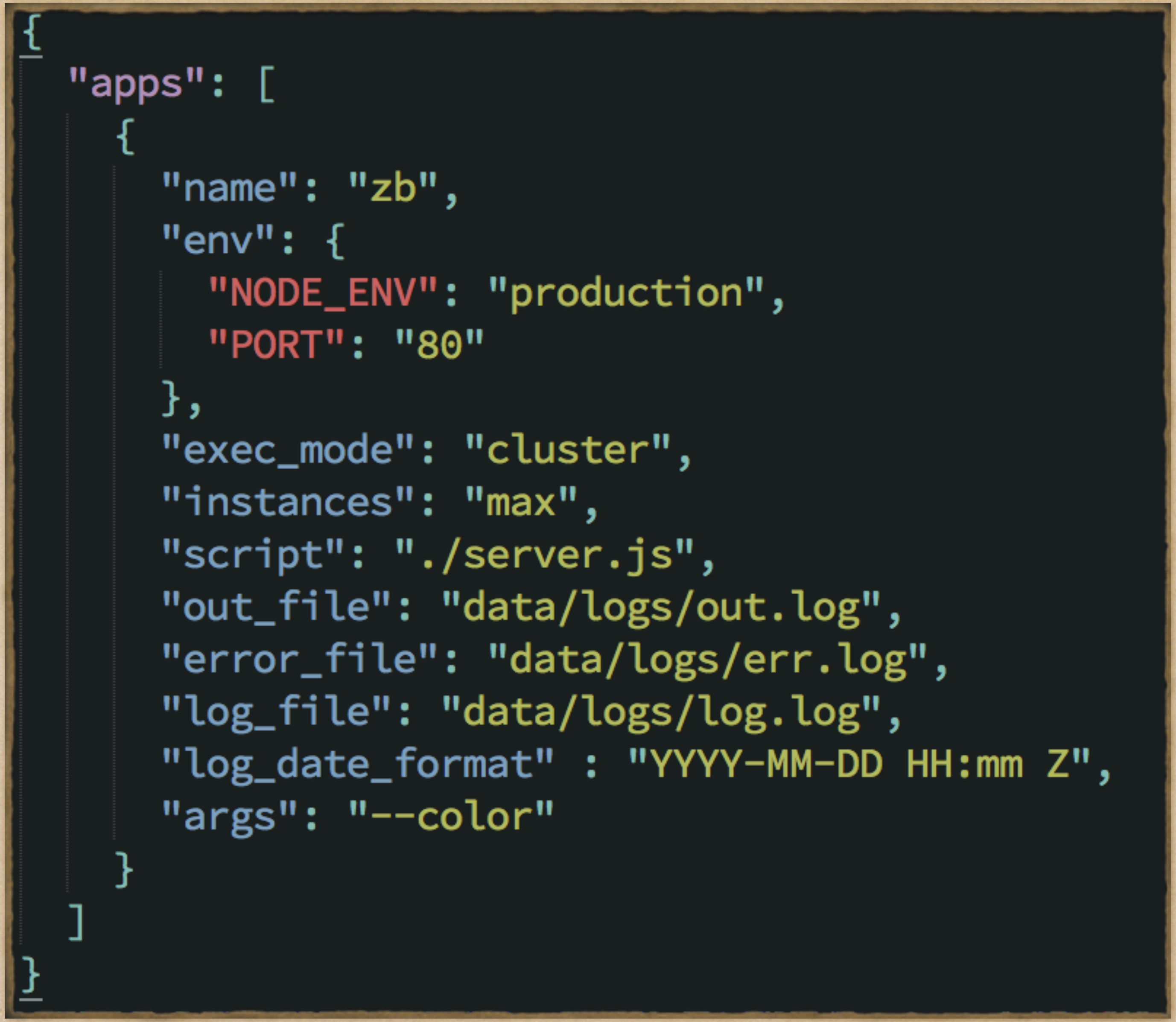
2.句柄共享(windows)


PM2
Cluster
master:80
browser
request


nginx LB
Cluster
:8081
:8082
:8083
:8084
nginx upstream
browser
request

关于PM2

- 实现守护进程
- 支持fork 和 cluster模式，cluster模式基于node cluster
- 自动重启
- 进程状态监控



```json
{
  "apps": [
    {
      "name": "zb",
      "env": {
        "NODE_ENV": "production",
        "PORT": "80"
      },
      "exec_mode": "cluster",
      "instances": "max",
      "script": "./server.js",
      "out_file": "data/logs/out.log",
      "error_file": "data/logs/err.log",
      "log_file": "data/logs/log.log",
      "log_date_format" : "YYYY-MM-DD HH:mm Z",
      "args": "--color"
    }
  ]
}
```

pm2 start server.json

异常进程

- 孤儿进程：爹被干掉了
- 僵尸进程：爹不给收尸😭

后续

- 完善监控和日志
- 对接服务化
- node在适合的场景创造更多价值

the end