操作系统及Linux内核

西安邮电大学

进程的概述

一个简单的C程序

```
#include <stdio.h>
int glob_a,glob_b=10;
int main()
{
static int local_val;
int i;
printf("glob_a=%d,glob_b=%d\nlo
cal_val=%d,i=%d\n",glob_a,glob_b
,local_val,i);
}
```

从程序到进程

从程序到进程

在Linux环境下对应的步骤：

1. gcc -S hello.c -o hello.s // 汇编
2. gcc -c hello.s -o hell.o // 编译
3. gcc hello.c -o hello // 链接
4. ./hello // 装载并执行

objdump -d hello //反汇编

从程序到进程


void X (int b) {
if(b == 1) {
...
int main() {
int a = 2;
X(a);
}
编译链接
文件头
代码
初始化数据
未初始化数据（BSS）
加载
共享库
栈
堆
初始化数据
代码
源程序
可执行文件
进程地址空间

感知进程

windows的任务管理器

linux下的top命令输出

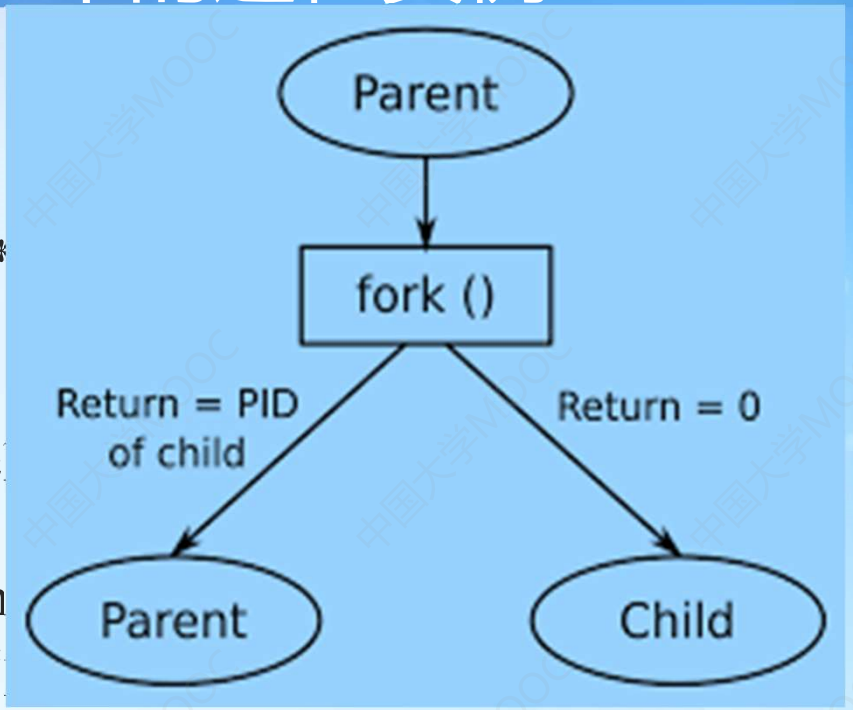
clj@cloudhhu:~

```
top - 22:01:11 up 208 days,  5:37,  2 users,  load average: 0.05, 0.04, 0.05
Tasks:  96 total,   1 running,  95 sleeping,   0 stopped,   0 zombie
%Cpu(s):  1.0 us,  0.3 sy,  0.0 ni, 97.0 id,  1.7 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :  1882892 total,    79472 free,   387428 used,  1415992 buff/cache
KiB Swap:  1049596 total,   841724 free,   207872 used.  1285728 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
29868 root      20   0  202496  67140   3704 S   0.7  3.6   8:03.12 YDService
22388 root      20   0  611508   8040   1928 S   0.3  0.4 195:21.02 barad_agent
    1 root      20   0   51760   2924   1856 S   0.0  0.2  59:31.28 systemd
    2 root      20   0       0      0      0 S   0.0  0.0   0:01.07 kthreadd
    3 root      20   0       0      0      0 S   0.0  0.0   5:21.89 ksoftirqd/0
    5 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/0:0H
    7 root      rt   0       0      0      0 S   0.0  0.0   0:00.00 migration/0
    8 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcu_bh
    9 root      20   0       0      0      0 S   0.0  0.0  50:40.86 rcu_sched
   10 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 lru-add-dra+
   11 root      rt   0       0      0      0 S   0.0  0.0   1:31.76 watchdog/0
   13 root      20   0       0      0      0 S   0.0  0.0   0:00.00 kdevtmpfs
   14 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 netns
   15 root      20   0       0      0      0 S   0.0  0.0   0:08.78 khungtaskd
   16 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 writeback
   17 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kintegrityd
   18 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 bioset
```

Unix/Linux中的进程实例

```
#include <stdio.h>
#include <unistd.h>
int num = 0;
int main(int argc, char*
    int pid;
    pid = fork();
    if(pid == 0){       /*c
        num = 1;
        printf("child_num
    }else if(pid > 0){  /*
        num = 2;
        printf("parent_nun
    }
}
```




```
[naomi@Host3 ~]$ ./fork
parent_num: 2
child_num: 1
```

什么是进程？

进程（强调并发性和动态性）

可以并发执行的程序在某个数据集合上的运行过程，是系统进行资源分配和调度的独立单位。

进程的结构

进程特征

结构性	程序+数据+PCB
动态性	创建，执行，调度，消亡
并发性	内存中有多个进程同时执行
独立性	是资源分配和调度的独立单位
异步性	各自独立运行，不知道谁先结束

程序与进程的区别

程序	进程
静态的	动态的
可长期保存	有生命周期
一个程序对应多个进程	一个进程可包含多个程序
代码+数据	代码+数据+进程控制块

进程的基本状态

就绪态

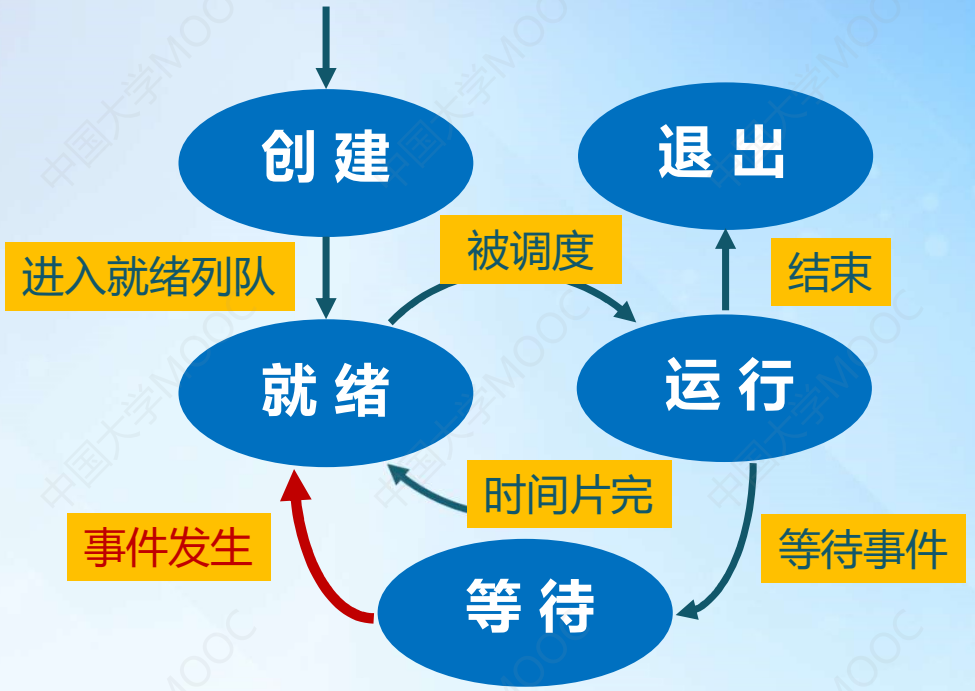
运行态

等待态

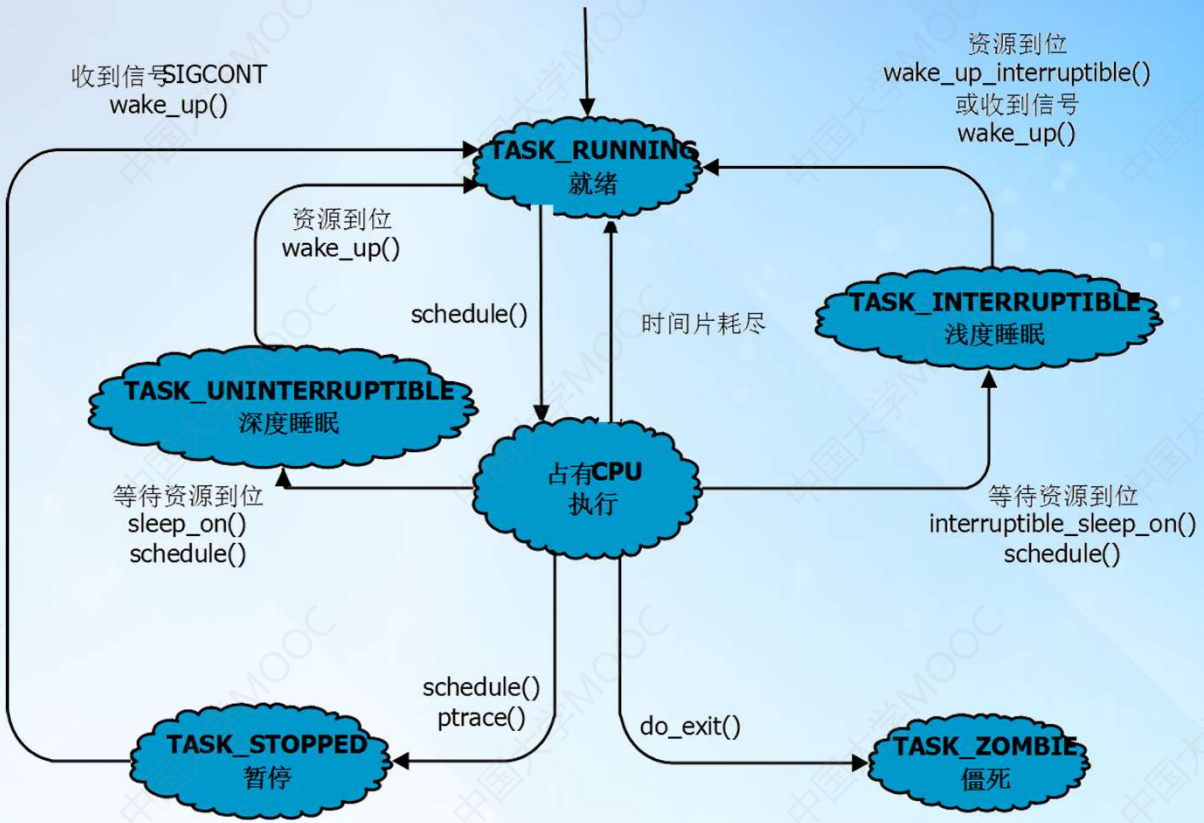
进程状态的转换


run run run
CPU
时间片到
进程调度
就绪队列
服务请求或等待I/O完成
等待队列
服务完成/事件到来
进 入
就 绪 队 列
PCB
PCB

进程状态的转换



Linux中进程状态及转换



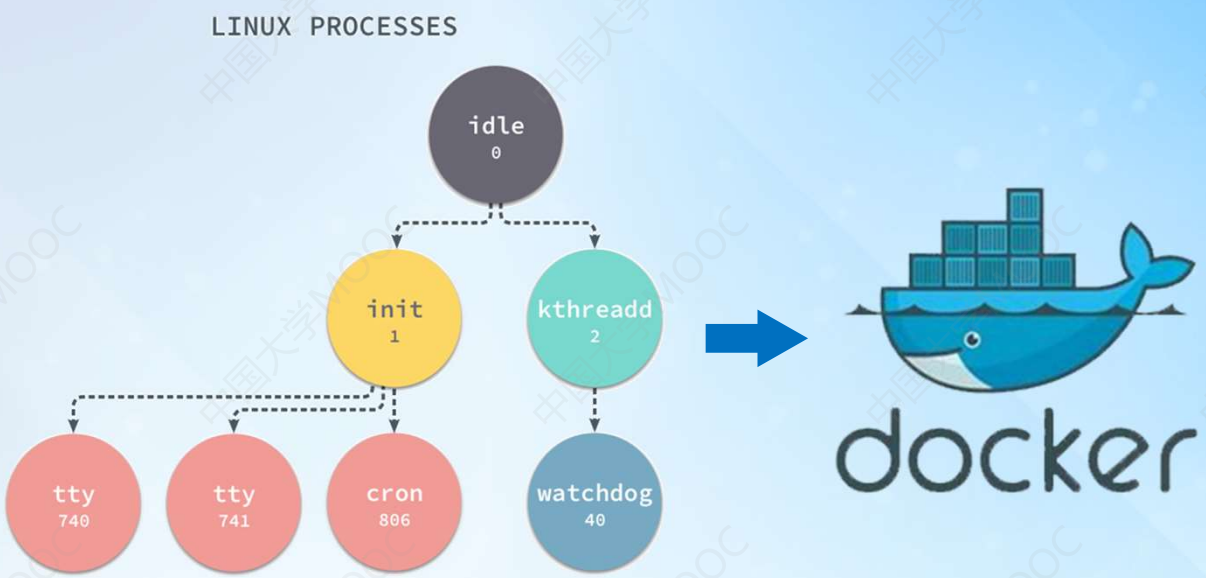
Linux中的进程状态

进程状态（在ps或top命令中看到的状态）	状态编码（在内核代码中定义的值）	状态的含义
R (running)	0	运行或将要运行
S (sleeping)	1	被中断而等待一个事件，可能会被一个信号激活
D (deep sleep)	2	被中断而等待一个事件，不会被信号激活
T (stopped)	4	由于任务的控制或者外部追踪而被终止
t (tracing stop)	8	
Z (zombie stop)	16	僵死，但是它的父进程尚未调用wait函数
X (dead)	32	死亡状态，这个状态永远也看不见

Linux内核源代码中进程状态的定义

```
/* Used in tsk->state: */
#define TASK_RUNNING                    0x0000
#define TASK_INTERRUPTIBLE              0x0001
#define TASK_UNINTERRUPTIBLE            0x0002
#define __TASK_STOPPED                  0x0004
#define __TASK_TRACED                   0x0008
/* Used in tsk->exit_state: */
#define EXIT_DEAD                       0x0010
#define EXIT_ZOMBIE                     0x0020
#define EXIT_TRACE                      (EXIT_ZOMBIE | EXIT_DEAD)
/* Used in tsk->state again: */
#define TASK_PARKED                     0x0040
#define TASK_DEAD                       0x0080
#define TASK_WAKEKILL                   0x0100
#define TASK_WAKING                     0x0200
#define TASK_NOLOAD                     0x0400
#define TASK_NEW                        0x0800
#define TASK_STATE_MAX                  0x1000
```

扩展--从进程到云计算中的容器



字体：中文：思源黑体≥24
英文：新罗马 ≥24

配色

层级 文件管理

特殊字体双击安装