

# 操作系统第三次展示

陆宇霄 同组：张延慈 张童

# 进程创建

寻找一个空的任务号

Task  
数组



```
find_empty_process start:task_nr:1
```

```
find_empty_process start:task_nr:2
```

```
find_empty_process start:task_nr:3
```

```
find_empty_process start:task_nr:2
```

```
find_empty_process start:task_nr:4
```

# 进程创建

进程空间写前验证，若页面只读，则  
执行共享检验和复制页面操作

[illegible][illegible]

# 进程创建

```
long state; //表示进程的状态, -1表示不可执行, 0表示可执行, >0表示停止
long counter; /* 运行时间片, 以jiffs递减计数 */
long priority; /* 运行优先数, 开始时, counter = priority, 值越大, 表示优先数越高, 等待时间越长. */
long signal; /* 信号. 是一组位图, 每一个bit代表一种信号. */
struct sigaction sigaction[32]; /* 信号响应的数据结构, 对应信号要执行的操作和标志信息 */
long blocked; /* 进程信号屏蔽码(对应信号位图) */
/* various fields */
int exit_code; /* 任务执行停止的退出码, 其父进程会取 */
unsigned long start_code, end_code, end_data, brk, start_stack; /* start_code代码段地址, end_code代码长度(byte),
end_data代码长度+数据长度(byte), brk总长度(byte), start_stack堆栈段地址 */
long pid, father, pgrp, session, leader; /* 进程号, 父进程号, 父进程组号, 会话号, 会话头(发起者) */
unsigned short uid, euid, suid; /* 用户id号, 有效用户id号, 保存用户id号 */
unsigned short gid, egid, sgid; /* 组标记号(组id), 有效组id, 保存的组id */
long alarm; /* 报警定时值(jiffs数) */
long utime, stime, cutime, cstime, start_time; /* 用户态运行时间(jiffs数),
系统态运行时间(jiffs数), 子进程用户态运行时间, 子进程系统态运行时间, 进程开始运行时刻 */
unsigned short used_math; /* 是否使用了协处理器 */
/* file system info */
int tty; /* 进程使用tty的子设备号. -1表示没有使用 */
unsigned short umask; /* 文件创建属性屏蔽位 */
struct m_inode * pwd; /* 当前工作目录i节点结构 */
struct m_inode * root; /* 根目录i节点结构 */
struct m_inode * executable; /* 执行文件i节点结构 */
unsigned long close_on_exec; /* 执行时关闭文件句柄位图标志. */
struct file * filp[NR_OPEN];
/* 文件结构指针表, 最多32项. 表项号即是文件描述符的值 */
struct desc_struct ldt[3];
/* 任务局部描述符表. 0-空, 1-CS段, 2-DS和SS段 */
struct tss_struct tss; /* 进程的任务状态段信息结构 */
```

31	16 15	0		
I/O 位图基地址		T	0x64	
		LDT 段选择符		0x60
		GS		0x5C
		FS		0x58
		DS		0x54
		SS		0x50
		CS		0x4C
		ES		0x48
		EDI		0x44
		ESI		0x40
		EBP		0x3C
		ESP		0x38
		EBX		0x34
		EDX		0x30
		ECX		0x2C
		EAX		0x28
		EFLAGS		0x24
		EIP		0x20
		CR3 (PDBR)		0x1C
		SS2		0x18
		ESP2		0x14
		SS1		0x10
		ESP1		0x0C
		SS0		0x08
		ESP0		0x04
		前一任务链接		0x00

保留位，设置为 0。

```
copy_process start:The input parameter:1 155272 4092 917504 23 30319 3 21992 33 23 23 23 26802 15 514 155248 23  
p->state:2 p->pid:1 p->father:0 p->counter:15 p->start_time:1 copy_mem start:
```

```
copy_process start:The input parameter:2 155272 4092 917504 23 30319 3 142016 26 23 23 23 27072 15 518 155216 23  
p->state:2 p->pid:2 p->father:1 p->counter:15 p->start_time:3 copy_mem start:
```

```
copy_process start:The input parameter:3 67108216 4092 1 23 30319 99274 12 309952 23 23 23 235695 15 582 67108204 23  
p->state:2 p->pid:3 p->father:2 p->counter:15 p->start_time:6 copy_mem start:
```

```
copy_process start:The input parameter:2 155272 4092 917504 23 30319 3 155216 0 23 23 23 27242 15 582 155216 23  
p->state:2 p->pid:4 p->father:1 p->counter:15 p->start_time:13 copy_mem start:
```

```
copy_process start:The input parameter:4 67104960 362145 76483 23 30319 366412 76462 362176 23 23 23 235695 15 582 67104948 23  
p->state:2 p->pid:5 p->father:4 p->counter:15 p->start_time:18 copy_mem start:
```

# 进程创建

复制内存

原代码段基址
原数据段基址
局部描述符中代码段描述符项中段限长
局部描述符中数据段描述符项中段限长
设置代码段描述符中基址域
设置数据段描述符中基址域

```
code_limit:655360
data_limit:655360
old_code_base:0
new_code_base:67108864
old_data_base:0
new_data_base:67108864
```

```
code_limit:655360
data_limit:655360
old_code_base:67108864
new_code_base:134217728
old_data_base:67108864
new_data_base:134217728
```

```
code_limit:655360
data_limit:655360
old_code_base:67108864
new_code_base:134217728
old_data_base:67108864
new_data_base:134217728
```

```
code_limit:266240
data_limit:67108864
old_code_base:134217728
new_code_base:268435456
old_data_base:134217728
new_data_base:268435456
```

```
code_limit:266240
data_limit:67108864
old_code_base:134217728
new_code_base:201326592
old_data_base:134217728
new_data_base:201326592
```

# 进程销毁

Sys_exit	Do_exit	tell_father	release
只是调用do_exit	设置子进程父指针	如果没找到父进程，就直接释放	扫描任务数组，找到指定任务，置空
	设置进程状态		重新调度
	通知父进程子进程要停止		
	重新调度		

```
sys_exit start:
do_exit start:
change task[2]'s father from 3 to pid_1
tell_father start:
tell father 1
do_exit call schedule~
waitpid call schedule~
realse start:
tast[2] is destroyed
tast[i] is availible now: 2 4 5 6 7 8 9 10 11 12
49 50 51 52 53 54 55 56 57 58 59 60 61 62 63
```

```
sys_exit start:
do_exit start:
tell_father start:
tell father 4
do_exit call schedule~
```

```
realse start:
tast[4] is destroyed
tast[i] is availible now: 4 5 6 7 8 9 10 11 12
50 51 52 53 54 55 56 57 58 59 60 61 62 63
```