L1 李子强 11510352

OS - speeial layer

=> including : Kernal - sys calls
Build-in library - C lib
Drivers
shell

Shell is terminal => Bash

Process in hierarchy

Sys call / lib calls

↓ stack FILO
↑ heap
Data segment
Consesmt
Text segment

Segmentation

File Allocation Table
FAT 32
-> 4 G = 2^{32}

NTFS
New Technology File system

Malloc ( )
⇒OS分配虚拟内存

L2

register / Cache / main memory

DMA => direet Memory Access

Kernel Data struct
- 单链表 · 双链表
- 环形链表 · 二叉搜索树
- 哈希表 · 比特映射

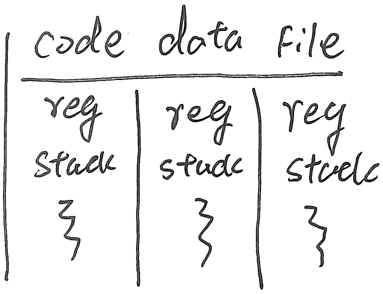
① Thread.
Single Unique execution context.
fully describe program state
[PC, Reg, Execution Flags, Stack]

② Address space. $2^{32} = 4G$
set of accessible addresses
& state associate with them.

Switch context. => PC, SP, reg.

③ Proteetion. (process)

1. Reliability
2. Security
3. Privacy
4. Fairness



④ Dual mode Kernel / User.

Mode swich
- Sys call
- interrupt 中断向量.
- Trap / Exception

L3 fork ( ) return child pid / 0
(in parent)

exec*( ) 清空原来地址空间
新命令return后不再回老命令。

Wait( ) 等待子进程 SIG CHILD
若子进程已返回，直接返回

WaitPid( )

init ( ) - first pid.

L4

PCB Process Control Block.

User time CPU time in User mode.

Sys Time ———— Kernel mode

fork ( )

In kernel
update PID / runtime. / Point to partent
In child. and list of children
in parent.

user space . All copy
opened File 0 stdin 1 stdout
2 stderr

exec ( )

user space :

Clear local var, Dynamic alloc mem

Reset Global var, Code / constants

exit ( )

free All In User space .

But Not in Kernel . still have entry In PCB ,

父进程接收 SIGCHIL 回收处理子进程

signal handlers 清理 返回 清理的pid.

未清理的 linux 会显示 defunct

Re-parenting by pid 1 => init
更新 parent poihe / list of children

L5

SIGSEGV / SIGCHLD

SIGINT / SIGTERM / SIGSTP
+C Kill +Z

SIGCONT | SIGKIL

同步 / 异步异常 SIGFPE 溢出

signal ( ) 信号注册 alarm ( ) 信号定时.

L6

Context switch

1. reg
2. address space (expensive)
[3. TLB / cache / buff reload]

Real 实际时间

Sys - Kernel mode

user - user mode

user ≥ sys bound by CPU

user < sys bound by I/O

PCB 用双链表管理

调度算法 1. Short Job first
2. round robin 3. Priority ~

wait Time : TASK (开始 - 到达) + 中间等

Turnround : TASK (完成 - 到达)

Turnround - wait = 实际运行

Preemptive 抢占式

L7

Inter process Communication
1. pipe / named pipe
2. signal POSIX
3. Message
4. share mem
5. semaphore
6. socket

Race Condition 竞争状态
Solution 须满足
1. Mutual exclusion 独一
2. Boundell waiting 有限时
3. Progress 一定有

解决方案 :
1. CS 直接屏蔽上下切换
2. Basic spin lock 忙
3. P Terson solution
4. Semaphore
5. POSIX Semaphore 阈 => post +1
6. mutex-lock wait -1

share obj . multi process , concurrency