C的代码区域

（低地址）(Low address)

代码区(coding memory area)

静态内存区（static memory area）：静态变量，全局变量，常量

Static memory area: static, const, global variable…. Release by OS

动态内存区（dynamic memory area）：堆（heap）（malloc（），free（））和栈（stack）（局部变量）

Dynamic memory area-heap: malloc() calloc() need programmer to release free() , stack: local variable release by OS.

（高地址）(High address)

堆和栈: （堆和栈这两种内存分配的方式），栈先入后出（可以用来保护现场），栈的内存分配和释放由系统完成，堆的内存分配和释放工作由程序员控制，所以栈的效率比较高。栈的大小远远小于堆的大小。堆是都是动态分配的，栈可以静态和动态分配。栈向下生长，堆向上生长。Malloc/free，new/delete , malloc(heap)/alloc(stack) for user, kmalloc/vmalloc for kernel

Memory for stack is assigned and released by OS, and memory for heap is assigned and released by programmer, so stack has more efficiency. The size of stack is smaller than heap. Heap is assigned dynamically whereas stack can both be assigned statically or dynamically. Stack be formed from high physical address to low physical address, heap be formed from low to high physical address.

堆分配和释放容易产生内存碎片：allocate and release by heap are prone to create memory fragmentation, which wastes resources.

静态变量和常量的区别？static和const关键字的作用。

For static, the address for this variable switch from stack area to static memory area, which means this variable can only be initialize one time.

For const, we can not do any modification to this variable.

Volatile和编译优化（当每次使用这个变量的时候，必须从原本的地址读数据而不是从临时的寄存器中）Each time this variable is used, data must be read from the original address rather than from the temporary register. For reject computer optimize the code while do compiling.

Define和typedef？Define do just copy procedure, typedef is give another name for a structure type.

三目运算符和define的副作用? Ternary operator

For define, only do copy/duplicate. For some conditions, if there will have a problem on operation priority after duplication, and the result probably different from the original one.

#define max(a, b) a\*a>b? a:b; c=max(a+1,b) c=a+1\*a+1…

条件编译（conditional compilation）#if，#ifdef，#ifndef，#error，#endif

指针函数和函数指针：指针函数是返回值是指针变量的函数，函数指针是指向函数的指针

Function pointer is a pointer head to one particular function

A function that returns a pointer.

位操作：1|1=1 1&1=1 AND OR NOR

Bit/bitwise operation. Like how to give bit3 a 1? How to give bit 3 a 0?

访问固定的内存位置:

We can first initialize a pointer to the physical memory address, and call by using asterisk (\*) pointer.

\*(char\*) (0x444444444)=’Z’; \*(int\*)(0x444444444)=0x4455aabb;

引用：给变量起别名

Give an alias to variable. Give an another name to variable.

中断和轮询

1、为啥要有中断（中断的重要性所在）  
和外部设备交互，双方的数据速率不对等（内部cpu比外部时间快），非常及时去响应外面主要任务。

External devices are slower than CPU. So if there is no interrupt, CPU would waste a lot of time waiting for external devices to match its speed with that of CPU. This decreases the efficiency of CPU.

2.中断和轮训及其差别  
中断是被动去解决外部的任务 Interruption is passive.  
轮训是主动定时的去查询和处理任务 polling is regular and active.  
NAPI是结合中断和轮训；  
3. 如何选择中断和轮训的方式？  
时间间隔确定和可控时，使用轮训；事件是随机时，使用中断；

Event’s interval is exact; random event  
4.中断类型--外部和内部中断external/peripheral interruption; internal interruption.  
外部和内部分别是cpu内部产生的事件，外部产生的事件和处理.

Generate and dispose

Events inside CPU, event outside CPU.   
5.中断应用场景和怎么被使用以及使用注意什么，什么是中断上下文（类似进程上下文）  
中断要求：现场保持，进入中断入口和任务处理，现场恢复；  
中断上下文（分步骤）：上文-收到和保存中断信息-不能睡眠和执行其他的系统调用，处理中断的紧急部分。下文-处理中断内容（比如收邮件和处理邮件分开为上下文，上下文的区别对应紧急响应，紧急处理）  
中断包含完整的中断上下文！

top half and bottom half for interrupt content:

top half (should as short as possible): the system receives interrupt information and saves break point at current process- the current process cannot hang up or sleep, and the system cannot execute other kinds of system call. Last, the system will handle the emergency part for this interruption.

Bottom half: Handle the non-emergency part for this interruption, like to execute interrupt process. And then back to previous break point.

Recode the breakpoint for current progress-> handle interrupt -> back to breakpoint proceed previous progress.

进程和线程

1.区别：进程是资源分配的单元，线程是是资源调度的单元  
进程好比一栋房子（可  
以有很多房间--线程，所以关系是1：n，一个线程仅属于一栋房子）；  
房间自己有独立空间，但是有公摊-公共空间(比如客厅)--共享地址空间  
2.进程和线程在通信方式  
线程之间有共享空间-距离”近“。通信方式多--13种：全局变量，锁，信号，邮箱，  
进程之间的通信，需要跨空间，方式较少，进程5种--socket，管道，共享内存，消息队列，信号量  
常考要对比这五种方式--不同的方式：  
3.为何说进程是资源分配（地址空间给房子的，分配管理地址是以房子为单位）  
为何说线程是资源调度的单位（在某个房间内执行任务，该忙完之后给另外一个房间，不同房间之间按照优先级竞争调度）  
4.执行效率，进程快还是线程快？（开放题目，系统是怎么根据任务级别去调度，资源节省的是什么）  
地址、数据指令：线程距离进切换快节省时间，进程是需要发现自己找的东西不在自己房子的所有房间内，需要通过比如管道去寻找另外其他的房子的房间内所有信息是否有东西，所有耗时大。  
两者的差异在于地址切换是开销；同在一个地址空间内不存在太多的地址切换；  
所以大部分线程是比进程更轻量级的处理；（当然，在那种一房子仅有一个房间，那么两种是没有本质差别的）  
5.线程开销低？能否尽量少用进程？  
代码健壮性，狡兔三窟，万一一个房子着火了就系统崩溃了；比如守护进程非常关键；  
房子独立，便于资源独立和保密资源；  
6.用进程还是线程的三个方面  
某个任务对当前的进程关系大不大？  
该任务 某崩溃了对原来的影响怎么样  
该任务的执行效率是否很重要--重要选择线程

用户态和内核态

User state and kernel state

区别（执行权限区分）  
（1）两者以资源操作权限为区分，执行权限不一样，相互协作，用户态无法调度内核态，内核态也不干涉用户态.

User state and kernel state have different privilege: kernel for level 0, user for level 3.  
（2）运行空间不一样：标准Linux0-3G用户态。3-4G内核态作为使用；用户态共享内核态.

They have different physical address: User for 0-3G, kernel for 3-4G.

两者不可以互相直接访问，如果访问会出现访问非法地址或内核崩溃！  
They cannot mutually access, if do, illegal address or kernel panic.  
内核态可以通过什么方式访问用户态？  
临时改变内核空间，3-4G编程0-4G改变，访问完恢复改变；

Change the kernel address temporarily from 3-4G to 0-4G, and recover after access.  
用户态正常工作是如何进入内核态？  
/系统调用（不是C的库调用）system call  
/中断（保存现场，进入中断上下文，上下文是处在内核态返回） interruption:   
/异常（会发信息给内核）exception: it will send signal to kernel  
2.两者如何通讯？（I/Ocontrol，sysctl动态修改内核参数，socket，procfs进程文件系统,sysfs读取和配置...）

Signal   
procfs 主要是输出只读数据，而 sysctl 输出的大部分信息是可写的  
netlink是Linux用户态与内核态通信最常用的一种方式，它可以用于内核与多种用户进程之间的消息传递系统，比如路由子系统，防火墙（Netfilter），ipsec 安全策略等等

锁

互斥锁自旋锁 信号量 读写锁

互斥锁和自旋锁：中断用哪个？为什么？自旋锁和互斥锁嵌套使用？？

条件变量+互斥锁：

读写锁（RCU）：读得多写得少。

死锁现象（锁的力度问题）：关键区域（的保护）和关键结构（效率小于健壮性）。两个或两个以上的执行序在执行过程中，因争夺资源而造成的一种互相等待的现象。

如何避免死锁现象？多个线程采用相同的加锁顺序。调用timeout参数。死锁检测（算法层面实现）

内存屏障：在执行下一条（读写）指令之前，把之前的（读写指令）指令执行完。（优化之后也许会有内存乱序访问问题）内存回写了吗？？

字节对齐byte alignment : 有无#pragma pack()？

The start address can be divided by byte the data has.

Bool char 1, short 2, long int float 4,long long double8, string 32.

同步synchronous和异步asynchronous:

Synchronous: Once the call is started, the caller must wait for the method(function) to complete before proceeding with subsequent methods.

Asynchronous: the caller no need to wait until method complete.

RTOS/NON-RTOS实时系统和非实时系统：1.区别（判断标准）实时系统：能够准确地预估当前任务被执行的时间的系统（例子：地铁）2.本质区别：任务调度方式有区别(Task scheduling method are different)。事实系统：可抢占性的任务调度方式(preemptive scheduling)（优先级高的任务先被执行并且不可以被其他的低优先级任务打断。） 非实时操作系统：一般情况下没有任务优先级的概念。

实时系统包含实时任务调度器：RTOS **including real-time task scheduler: 1. Preemptive scheduling 2. Time slice scheduling (round-robin timeout)**

1.Real-time system can accurately estimate when the current task will be executed.

2.Real-time system has task priority, preemptive task mode.

Preemptive scheduling: the current task will always run until been break by another task with higher priority or meet blocking API (Application Programming Interface), and Higher priority tasks are executed first and cannot be interrupted by other lower priority tasks.

Time slice scheduling (RR): every task have the same priority. Every task will be assigned with the same time period control cpu, when times up, it will switch to next task.

看门狗：监控模块（软件模块或者硬件模块）没有被reset(系统没有异常发生)，达到某一个条件的话，就会执行某一个操作（up某一个程序或者重启整个设备）。

Watchdog: if a condition is met, an operation is performed.

TCP和UDP的区别: 0. TCP是面向连接的协议，UDP是无连接的协议。 1. TCP (可靠的传输协议：双方互相确认) 。UDP：发个东西过去就不管了，不提供任何交付的保证（传出去就OK了）2. 发送消耗（TCP大 UDP小）3. TCP速度慢, UDP速度快。

TCP/IP三次握手四次挥手？

TFO（TCP快速打开）的原理？

什么是SYN flood攻击？什么是半链接队列？

TCP协议中时间戳的作用？ 精确计算出RTT

RTO（Retransmission Timeout）是如何计算的？RTT (round-trip time) （往返延时）波动比较小的情况下：取RTT平均值。引入（平滑往返时间SRTT）

TCP如何实现流量控制？发送缓存区，接收缓存区。如何获悉对方的接收缓存区满了？接收窗口 (RWND)（TCP报头的窗口部分）Chart, box and whisker chart

Description automatically generated

滑动窗口

Application

Description automatically generated with medium confidence

TCP Keep-alive ？ 作用是探测对端的链接有没有失效。定时（7200s）发送数据探测包，（替代：应用层心跳机制）

TCP端口号？ 源端口和目标端口。2^16 65536（0-65535）个端口号（两个字节16位二进制数）

Table

Description automatically generated

ARP 攻击Text

Description automatically generated

ARP欺骗（监听）

Text, letter

Description automatically generated

Text

Description automatically generated

防御？ 静态APR绑定

Text, letter

Description automatically generated

输入url之后，浏览器有什么变化？(what happen when we enter URL?)

1.域名补全

2.DNS解析(domain name service)

3.TCP/IP链接（包括各种层的各种协议）

Application layer, socket, transmission layer, network layer, data link layer, physical layer.

4.服务器处理请求并且返回数据包

5.浏览器渲染

Table

Description automatically generated with medium confidence