嵌入式填空题和简答题:

第一章 绪论

1、IEEE定义:嵌入式系统是控制、监视或者辅助设备、机器和车间运行的装置。

2、一般定义:嵌入式系统是嵌入到计算机体系中，用于执行独立功能的专用计算机系统。

以应用为核心，以计算机，通信，控制等技术为基础，采用可剪裁软硬件，适用于对功能，可靠性，成本、体积功耗有严格要求的专用计算机系统。

3、简单的嵌入式系统:仅有执行单一功能的能力，在唯一的ROM中仅有实现单一功能的控制程序，无微型操作系统

4、嵌入式系统三要素:嵌入性，专用性，计算机系统

5、嵌入式系统与桌面通用系统的区别:

嵌入式系统用于执行专用而确定的任务，而桌面通用系统需支持大量的、需求多样的应用程序

嵌入式系统对实时性有较高的要求

嵌入式系统对可靠性要求较高

嵌入式系统有功耗约束

嵌入式系统内核小，可用资源少

嵌入式系统的开发需要专用工具和特殊方法

6、嵌入式系统的核心技术:3C（通信，计算机，消费电子）

7、嵌入式系统结构:嵌入式系统由嵌入式处理器，外围硬件设备，嵌入式操作系统(可选)，用户应用软件系统组成

8、嵌入式系统分类:

按嵌入式微处理器字长:4/8/16/32/64

按实时性:非实时，软实时，硬实时

按复杂程度分:小型系统，中型系统，复杂系统

按系统形态分:芯片级，板级，设备级

第二章 ARM相关

1、计算机体系结构分为:冯诺依曼结构、哈佛结构

2、RISC:精简指令集计算机系统 采用Load/Store结构

3、CISC与RISC对比:

指令系统:指令多/少

执行时间:执行时间长/短

编码长度:指令长度不固定，指令长度固定

寻址方式:寻址方式多样，寻址方式简单

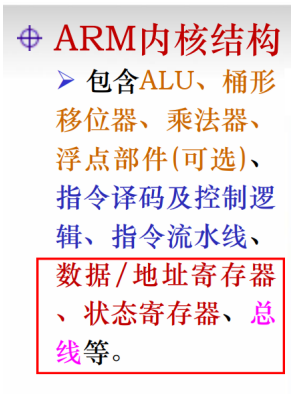
操作:可对存储器和寄存器操作，只能对寄存器操作

编译:难以使用优化编译器，采用优化编译技术

4、ARM的含义 :一个公司的名字，一类微处理器的通称，一种技术的名字

5、ARM微处理结构:ARMCPU + 外部设备 = ARM芯片

6、



7、运行模式:

用户模式:ARM微处理器正常的工作模式

FIQ：用于高速数据传输和通道处理

IRQ：用户通常的中断处理

管理模式：操作系统使用的保护模式，用于处理软件中断

数据访问中止模式：用于虚拟存储器和存储器的保护

未定义指令模式：支持硬件协处理的软件仿真

系统模式:**运行具有特权的操作系统任务**

8、R14两种特殊功能：在每种模式下用于保存子程序的返回地址；发生异常时用于保存异常处理程序后的返回地址

9、存储管理单元MMU：用于虚拟地址与物理地址的转换、存储器访问权限控制、设置虚拟存储空间的缓冲特性

10、虚拟存储管理:分页式、分段式、段页式

11、基于TLB的地址转换过程:微处理器访问主存时首先根据虚拟地址在TLB中查找对应的地址转换条目，若命中则直接进行转换；微处理器继续在位于主存中的页表中进行查找，找到后将对应的地址转换条目放入TLB中，若TLB已满则使用替换算法。

12、虚拟存储空间到物理存储空间的映射是以**存储块**为单位进行的

13、CP15（16个）:系统控制协处理器，作用：对MMU进行配置完成存储系统管理

14、平板地址映射：物理地址 = 虚拟地址

15、使能MMU前的注意事项：在主存中建立0号页表，同时CP15中各相关寄存器完成初始化；若设置的物理空间和虚拟存储空间大小不等，在禁止/使能MMU前需要清空Cache中对应地址转换条目

16、域：段/大页/小页的集合

17、存储访问失效:地址对齐失效、地址变换失效、域控制失效、访问权限控制失效

18、存储器可以中止的三种访问:Cache内容预取、非缓冲存储器的访问、页表访问

19、异常处理过程：当异常发生时，系统保存当前处理器的状态，然后跳转到相应异常处理程序处执行；当异常处理结束后恢复现场并返回源程序发生异常的下一条指令执行

20、异常处理的步骤：将下一条指令的地址放入链接寄存器LR中，用于当异常处理程序执行结束后的正确返回 将CPSR中的内容复制到对应模式下的SPSR寄存器中，根据运行模式强制改变CPSR中的模式位；强制PC从异常地址中取出下一条指令并执行从而跳转到异常处理程序处并设置中断禁止位。

21、异常返回的步骤：将链接寄存器LR的值送入PC;将SPSR中的内容恢复到CPSR中；若异常处理过程中设置了中断禁止位需要清除

第三章 ARM 指令相关

1、指令：计算机控制各功能部件协调动作的命令

2、指令系统：微处理器所能执行全部指令的集合【不同微处理器有不同的指令系统，指令由硬件直接识别并执行】

3、机器语言语句：能被微处理器识别的二进制编码，是指令在计算机中的表示形式

4、汇编语言指令：机器指令的符号化表示形式 (由操作码和地址码组成)

5、操作码：指示指令要执行的具体操作，用**指令助记符**表示

6、寻址方式:

寄存器寻址、立即寻址、寄存器偏移寻址、寄存器间接寻址、基址寻址、多寄存器寻址、堆栈寻址、相对寻址 【对PC的操作】

7、ARM中两种实现程序流程的跳转方法：使用专门的跳转指令可以实现向前或向后32MB地址空间的跳转、直接向程序计数器中PC中写入跳转值可以实现4GB地址空间的跳转

第四章 存储器

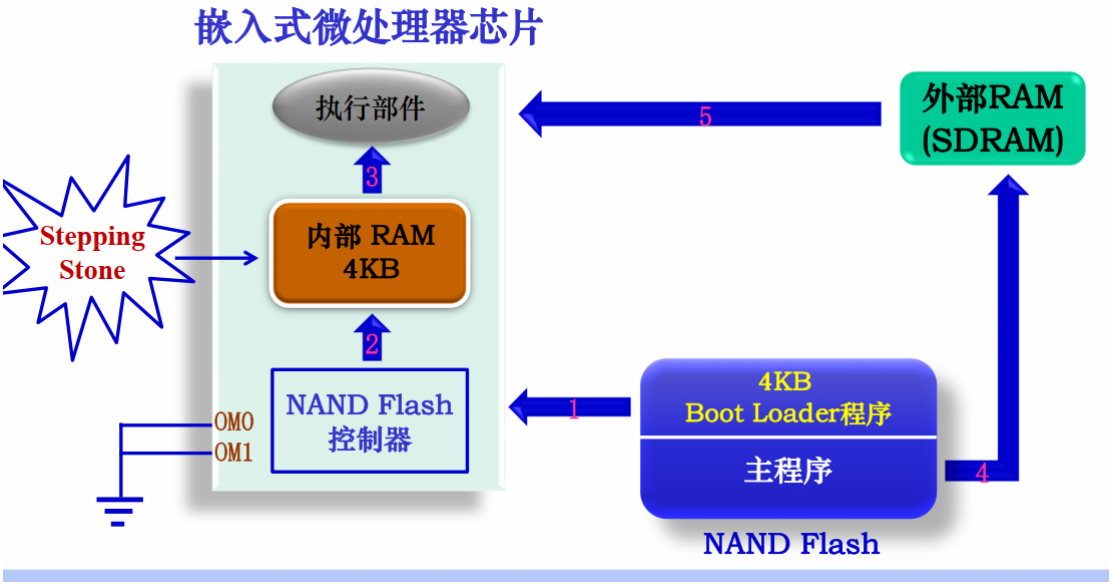
1、存储器类型分类：按所在位置分类(主存储器,外部存储器);按存储介质分类(半导体存储器,光盘存储器,磁表面存储器);按存取方式分类(随机存取存储器RAM,顺序存取存储器(SAM),直接存取存储器(DAM),只读存储器(ROM),闪速存储器(Flash Memory))

2、嵌入式存储器结构：内部寄存器(存放数据，加速指令的执行速度)，高速缓冲存储器(存放欲执行的数据和指令，提高指令的执行速度)，主存储器(存放欲执行的数据和指令)，主存储器(存放固化的系统程序，运行初始化程序),外部存储器(存放大量数据)

3、s3c2440存储器**可寻址的外部空间为1GB**

4、Bank0-5:可连接SRAM,具有SDRAM接口特性的ROM

5、Bank6、7：可连接SRAM,具有SDRAM接口特性的ROM以及SDRAM



使用NAND Flash引导程序的启动:当微处理器使用NAND Flash引导程序启动时，首先将位于NAND Flash内部大小为4KB的BootLoader程序通过NAND Flash控制器映射到内部RAM中，RAM中对应的位置称为Stepping Stone,接着将位于内部RAM中程序通过执行部件进行运行，之后将NAND Flash中剩余程序送入外部RAM中，通过执行部件进行执行操作

7、Nor Flash与NAND Flash的特点与对比

Nor Flash

读取速度快，具有芯片内执行XIP特性；写入速度慢，单位体积内容量小价格高；擦写次数越10万次；**带有SRAM接口**，与**微处理器**连接方便，便于数据存取；适用于存储固化的系统启动引导代码，操作系统代码，应用**程序**代码；通常配置到Bank0，**当系统上电或复位后**从其内取出指令并开始执行

NAND Flash

擦除和**写入**速度快，单位体积内数据存储密度大，价格相对便宜；使用复杂的I/O接口电路和存储器管理操作【以页为最小单位进行读写，以块为最小单位进行擦除】；擦写次数约100万次；适用于存储大量用户数据和程序代码；支持自动启动引导

8、NAND Flash两种工作模式:自动引导模式；普通闪存模式

第五章 常用外围设备

1、通用输入输出口:

GPA 23引脚 2种功能 1输出 2 输出外接**主**存储器的地址信号和存储块选择信号以及外接NAND Flash 存储器控制信号

GPB 11引脚 2种功能 1输入/输出 2 DMA,总线请求信号和应答信号以及各种时钟信号

GPC 16引脚 2种功能 1输入/输出 2 LCD显示器数据信号与控制信号

GPD 16引脚 3种功能 1输入/输出 2 LCD显示器数据信号 3 SPI总线控制信号

GPE 16引脚 3种功能 1输入/输出 2 SPI/I^2C/I^2S 总线和**SD存储器**的控制信号 3 AC'97音频接口的控制信号

GPF 8引脚 2种功能 1输入/输出 2 中断请求信号

GPG 16引脚 3种功能 1输入/输出 2 中断请求信号 3 LCD显示器控制信号以及SPI总线控制信号

GPH 11引脚 3种功能 1输入/输出 2**异步串行接口**UART 3异步串行通信接口UART(仅2个)

GPJ　13引脚 2种功能 1 输入/输出 2 摄像头接口的数据信号和控制信号

2、上拉寄存器作用GPBUP-GPJUP：简化嵌入式系统的硬件电路设计，对于需要上拉电路的IO端口无需设计上拉电阻

3、驱动强度控制寄存器GSC0-GSC1：用于设置ARM微处理器芯片数据总线、地址总线以及部分作为第二/第三功能使用的GPIO引脚的驱动电流大小

4、中断概念：微处理在执行正常程序过程中，因某事发生收到来自外围部件的请求信号

5、中断作用：并行处理，实时处理，故障处理

6、由**芯片外部中断请求引脚和内部外设触发的是**外部中断请求和快速中断请求

7、借助**中断控制器**接收并管理60个中断源发出的中断请求信号

8、中断控制器的功能：外部中断请求信号管理、中断模式设定、中断请求信号标记、中断屏蔽设定、中断优先级管理、中断服务标记

9、当多个中断源的中断请求信号同时有效时，需要通过**中断优先级**来确定对这些中断请求的服务顺序

10、**中断控制器借助优先级裁决器**实现对32个一级中断源的优先级判决

11、中断挂起寄存器**只对IRQ模式有效**

12、外部中断滤波寄存器EINTFLT：用于设定8个外部中断源EINT16 - EINT23的滤波器时钟信号来源和滤波宽度

13、源挂起寄存器：用于标记32个一级中断源的中断请求信号是否被**触发**

14、**中断挂起寄存器**：用于标记32个一级中断源的中断请求是否**即将或正在被微处理器服务**

15、时钟部件：时钟控制模块

16、定时部件：定时器、实时时钟、看门狗定时器

17、三种内部时钟信号：FCLK(供微处理器内核使用的时钟信号)、HCLK(供高性能总线AHB使用的时钟信号)、PCLK(供外部总线APB使用的时钟信号)

18、两种时钟源：晶体振荡器、外部时钟源

19、定时器：定时，计数，脉宽调制(PWM)

20、对于写入操作 要记住ldr r0，= 后面是一个16进制的数 该操作等同于mov r0,一个数 但是记住ldr 不作为伪指令使用时不可以使用mov指令与之对应

21、实时时钟RTC 表示为8位BCD码因此读出和写入时是以字节为单位的

22、看门狗定时器：一种用于噪声或系统错误引起故障时恢复微处理器操作的定时器

23、16位内部定时器，当计数值为0时通过触发中断服务可以激活128个PCLK时钟周期的内部复位信号

24、**只能**使用PCLK作为源输入时钟信号，且**无对外输出引脚**；系统处于嵌入式ICE模式时WDT自动关闭

25、

△计数值=所需时间间隔÷T_watchdog = 所需时间间隔×F_watchdog

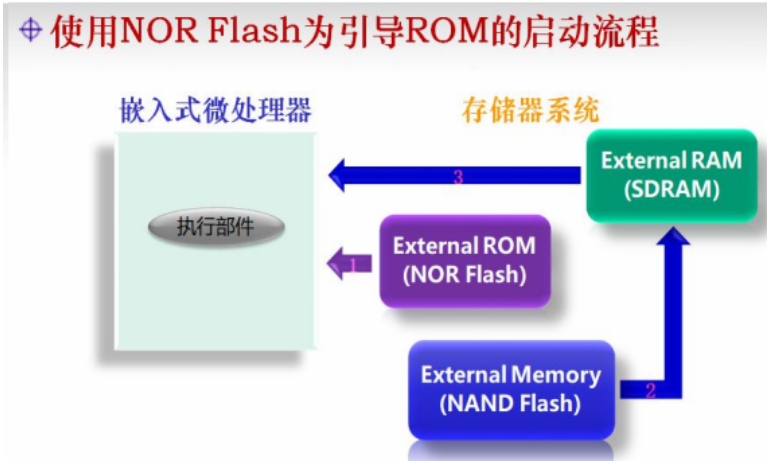
26、

- **不同于普通定时器，WTDAT的值在WDT初始使能时，不能自动装载到WTCNT中，所以必须要给WTCNT设定一个初始值。**

27、看门狗定时器工作原理：系统初始使能时由WTCNT中开始减1操作，当计数值即将减少到0时，系统对定时器进行重载操作，将WTDAT中的值装载到WTDAT中，每次重复此种操作俗称喂狗，若某次因系统故障导致系统无法重载计数值则当看门狗定时器减少至零后会触发复位使得系统重启。

第六章 Linux相关

1、



通过ROM直接由执行部件执行引导装载程序，之后将NAND Flash中程序送入外部RAM中由执行部件进行执行

2、Bootloader（引导装载程序）:是嵌入式系统上电后，**操作系统内核**运行之前运行的第一段程序

3、用途：为最终调用操作系统内核准备好正确的软硬件环境

4、Bootloader工作模式:启动加载模式；下载模式

启动加载模式：准备好操作系统内核运行时所需的环境和参数；从目标机某个固态存储设备上将操作系统装载到RAM中；在RAM中运行操作系统内核

下载模式：目标机上的Bootloader通过串口或网络等通信手段从主机下载文件到目标机的RAM中然后写入目标机上的Flash类型固态存储设备中。

5、Stage1:硬件设备初始化；为加载Bootloader的stage2准备RAM空间；拷贝Bootloader的stage2到RAM空间中；设置堆栈；跳转到stage2的C程序入口处

6、Stage2:初始化本阶段用到的硬件设备；检测系统内存映射；将内核映像和根文件系统映像从Flash存储器中读入到RAM空间中；为内核设置启动参数；调用内核

7、Linux：**目标是创建一套完全自由的操作系统**

8、Linux：GNU/Linux

9、典型Linux发行版包括：Linux内核、GNU程序库和工具、命令行Shell、图形界面和桌面环境、应用软件

10、内核：运行程序和管理设备的核心，用于管理内存、CPU和其它相关组件

11、系统调用控制：**内核的核心行为**

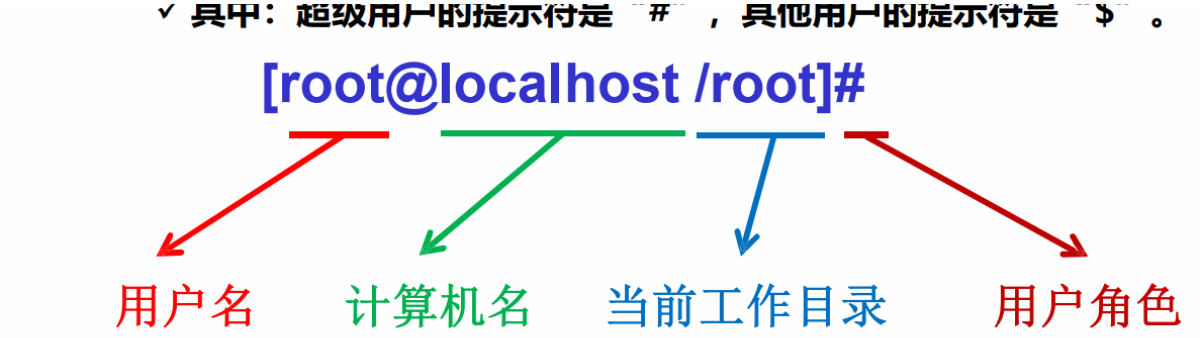
12、Linux内核中五个主要的**子系统**：进程调度、进程间通信、内存管理、虚拟文件系统、网络接口

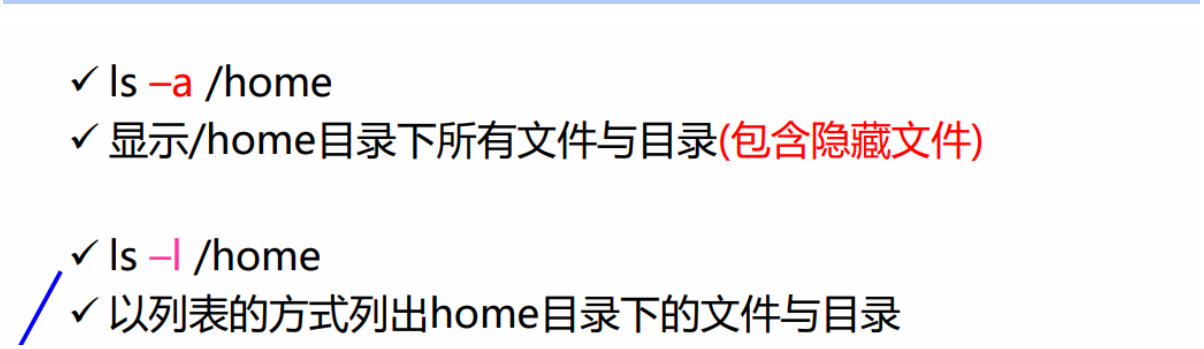
13、内核烧写方法：通过串口、通过MicroUSB接口、通过SD存储卡、通过网络、通过JTAG接口

14、Linux与Windows文件系统对比：Windows文件系统只负责文件存储、Linux文件系统负责管理所有软硬件资源

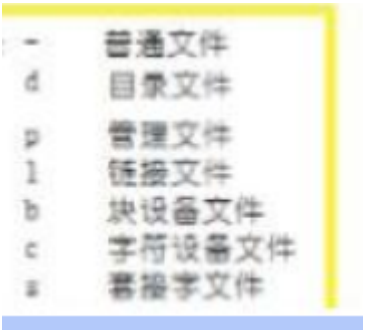
15、/etc:存放配置文件 /dev存放外设 /opt第三方软件 /usr应用程序安装目录

16、三种基本文件类型：普通文件、目录文件、设备文件





17、对文件或目录进行访问的三种用户类型：文件所有者，与所有者同组的用户、其他用户

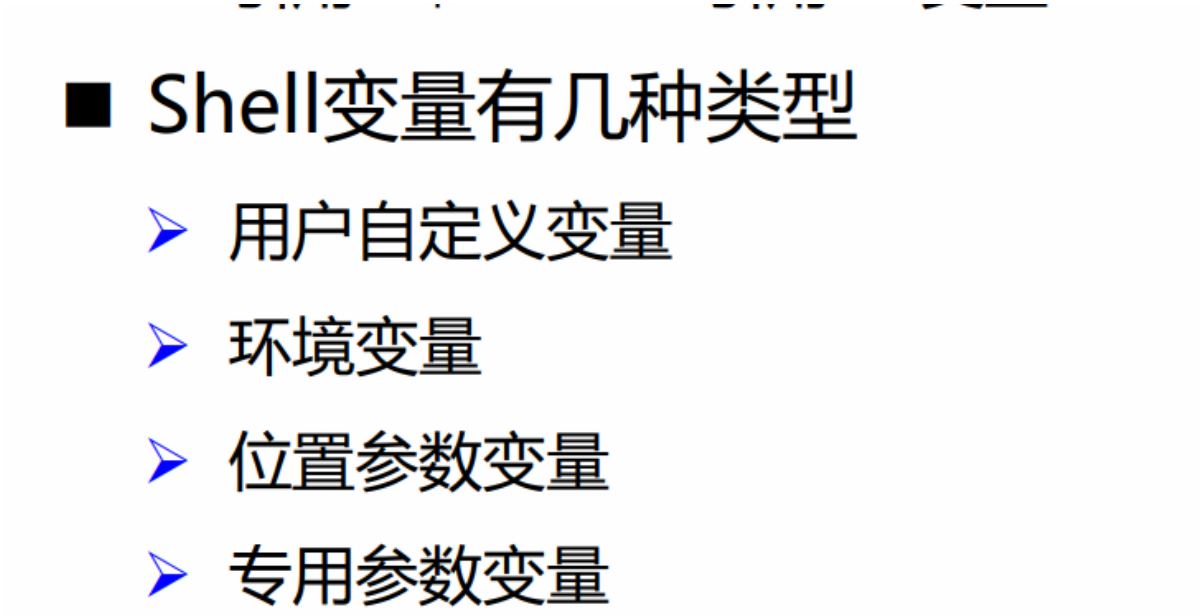


18 u文件所有者 g与所有者同组 o其它用户 a全部

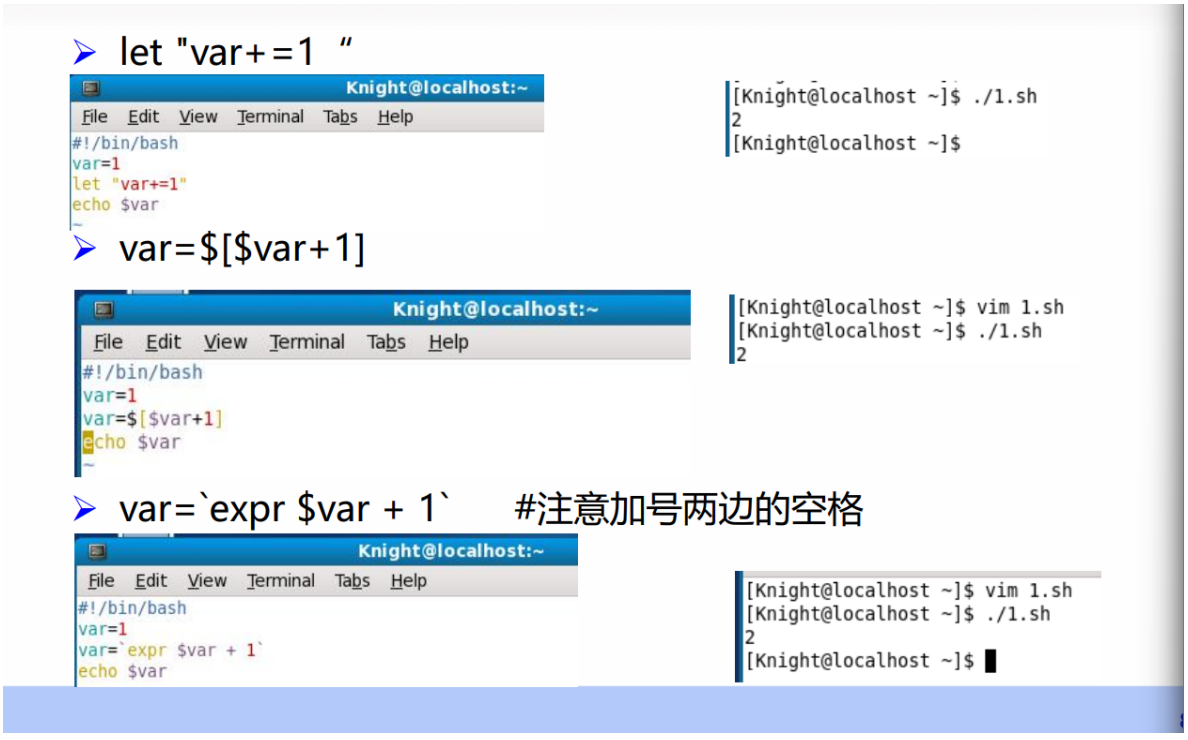
19、shell一般结构：shell类型，函数，主过程

20、程序编译和运行过程：编辑文件、保存文件、为文件赋予可执行权限、运行及排错

21、



22、



23、-o输出文件 -e预处理 -s编译 -c汇编

24、gcc使用与开发步骤：使用gcc编译代码、生成预处理文件、生成汇编文件、生成目标文件、生成可执行文件

25、结构：目标：依赖

命令