- float 算术运算符 type, operator, expression - 关系运算符 逻辑运算符 —— || 运算符 位运算符 ~(一元运算符,按位取反) 赋值运算符 —— %= =&= 条件运算符 —— ?: if-else switch-case contral flow do-while break,continue goto, label external variable and function — 所有用到的源文件只定义一次,可以多次声明 与全局变量的区别在于 1. 可以隐藏在其他资源(例如文件)中的可见性。 2.存储在静态变量区,具有持久性。 static variable 可用于internal变量,使其具有持久性 static variable and function static function — 隐藏可见性(只有定义函数的文件可以访问该函数) 该关键字建议编译器将变量存储在机器的寄存器中,但是机器可 register variable 以拒绝。 external and static variable 默认初始化为0,并且只能使用 constant expression来初始化。(不能使用其他变量或者函数 functions, program structure 调用来初始化) initialization automatic and register variable 默认初始化undefined recursion #include "filename" — 一般include自己项目中的其他文件时使用 file inclusion #include <filename> — 一般include库文件时使用 #define name replaced_text 还可以通过#undef name来消除macro macro substitution C preprocessor #用来进行原始字符串替换, 不展开。 https://zhuanlan.zhihu.com/p/344240420 & ##用来进行宏参数拼接。 #if SYSTEM == SYSV #define HDR "sysv.h" #elif SYSTEM == BSD #if expression #define HDR "bsd.h" #elif expression #elif SYSTEM == MSDOS conditional inclusion #define HDR "msdos.h" #endif #define HDR "default.h" #endif #include HDR 定义时使用*,type * pointer ─ 提取地址时使用&, pointer = &a pointer 访问pointer指向的元素时使用*,*pointer=x pointer加数值表示偏移,具体的偏移数值由指针所指的数据字节 长度和偏移量共同决定。 数组名就是一个指针 array 用{}初始化 区别:pointer是变量,可以赋值(作为左操作数)。数组名是地 址常量,不可以赋值。 ┍ pointer不可以与integer交换(0除外,但更建议用NULL)。 - pointer相减+1为元素数量 pointer数组的定义: type *pointer[length] · argc 参数数目 argv各个字符串参数的pointer数组。argv[0]是命令行程序名。 - 命令行参数的argc(参数数目)和argv() 有效参数从argv[1]-argv[argc-1]。要求argv[argc]为NULL。 用到以上两个参数的函数形式为: type func(int argc, char *argv[]) pointers, arrays int readlines(char *lineptr[], int nlines);
void writelines(char *lineptr[], int nlines); int (*)(void*, void*) func_name void qsort(void *lineptr[], int left, int right, int (*comp)(void *, void *));
int numcmp(char *, char *); 其中int (*)(void*, void*)为函数类型的指针前缀。与int * 本质上 是相同的。函数名在汇编中其实也是一个地址。 /* sort input lines */ main(int argc, char *argv[]) 并且将在函数指针前缀中将参数的指针类型置为void,则可以在 具体的函数中进行指针类型转换,达到不同的指针类型均可以处 if (argc > 1 && strcmp(argv[1], "-n") == 0) 理void类型指针的目的。 - 函数与pointer —— 函数名可以作为pointer参数传入 int i, last;
void swap(void *v[], int, int); 在调用时需要通过*得到具体的函数入口位置 ()会被翻译为CALL和IRET,通过这种方式完成函数调用 int *f() — 一个函数声明,该函数返回int 指针类型 🖊 inf (*pf) () —— 一个指针的声明,该指针指向的类型为返回int类型的一个函数 http://www.unixwiz.net/techtips/reading-cdecl.html & complicated declaration ✓ 1. x首先向右结合()得到x(),说明x是一个函数 ✓ 2. x()向左结合*得到*x(),说明这个函数的返回值是一个指针 具体分析复杂声明的方式 / 3. *x()向右结合[]得到*x()[],说明第2步最后的指针指向的是数 char (*(*x())[]) () — 4. *x()[]向左结合*得到*(*x())[],说明第3步最后得到的数组中 的元素是指针 5. *(*x())[]向右结合()得到(*(*x())[])(),说明第4步最后得到的指 针指向一个函数 []和()的优先级比*高,结合原则为优先向右结合,碰到必须向左 结合时向左结合(比如碰到作为整体的括号)。 6. (*(*x())[])()向左结合char得到char (*(*x())[]) (),说明第5步 最后得到的函数返回char类型 其他例子待补充 结构体可以作为函数的返回值 定义 — 语法: struct name {...}; · 定义同时声明 —— 语法:struct [name] {...} variable_i; —— name可以没有,如果后续只使用variable_i的话 / 定义、声明、访问变量 声明时需带上struct关键字,声明时若不赋值则不分配存储空 声明 — 语法: struct name variable_i; variable_i.member 声明 —— 语法: struct name *pointer 结构体指针 访问变量 — (*pointer).member pointer->member struct tnode {char *word; int count; 🧪 结构体自引用(使用在树或者链表结构) —— 举例 struct tnode *left; struct tnode *right; sturctures 常规类型语法 — typedef basic_type new_type 这种语法实际上是将new_type作为basic_type的指针的别名。 而不是将*new_type作为basic_type的别名。可以参考 pointer,array章节的复杂声明分析部分,从new_type开始向左 结合。 typedef — 指针类型语法 —— typedef basic_type *new_type -函数类型语法 — typedef int (*pf) () — pf为返回值为int类型的函数的指针的别名 union name{ type_1 name_1; 在内存中的本质:根据需要的最大存储空间的类型进行存储分配。然后根据具体的类型进行访存(指针类型转换)。 定义 —— . type_n name_n; union 常规类型 — union_name.member union_pointer->member (*union_pointer).member The C programming Language 用处: 以位为单位进行数据定义 bit-fields 说明:每个内部数据的说明虽然是unsigned int,但是实际分配 的数据只有1 bit. struct { unsigned int is_keyword:1; _unsigned int is_extern:1; unsigned int is_static:1; 赋值与测试操作: flag.is_extern=1
if flag.is_extern==1 } flag; int getchar(void) - int putchar(int) — 返回值输出字符的ASCII码值 %d 十进制整数 · %6d 长度显示至少为6位的整数 - %f 十进制小数 - %6.2f 长度显示至少为6位,小数点后为2位的小数 格式: printf (char *format, arg1, arg2, ...) %o 八进制 %x 十六进制 %c 字符 ─ %s 字符串 %开头 / [-]有-号表示左对齐,无-号表示右对齐 - format定义 ── number:表示最小宽度,可以大于该值 .隔离最小数据宽度与精度 number:表示精度 printf函数格式化 TABLE 7-1. BASIC PRINTF CONVERSIONS ARGUMENT TYPE; PRINTED AS int; decimal number.

int; unsigned octal number (without a leading zero).

int; unsigned hexadecimal number (without a leading 0x or 02), using abcdef or ABCDEF for 10, ..., 15.

int; unsigned hexadecimal number.

int; unsigned decimal number.

int; unsigned decimal number.

char *; print characters from the string until a '\0' or the number of characters given by the precision.

double; [-]m.dddddd, where the number of d's is given by the precision (default 6).

double; [-]m.ddddddd ±xx or [-]m.dddddd ±xx, where the number of d's is given by the precision (default 6).

double; use % or %E if the exponent is less than -4 or greater than or equal to the precision; otherwise use %E. Trailing zeros and a trailing decimal point are not printed.

void *; pointer (implementation-dependent representation).

no argument is converted; print a %. đ, i format字符与对应的数据类型 -:%s: :hello, world: :%10s: :hello, world: :%.10s: :hello, wor: :%-10s: :hello, world: format举例 —— :%.15s: :hello, world: :%-15s: :hello, world : :%15.10s: : hello, wor: :%-15.10s: :hello, wor : standard input and output 特殊情况: 宽度和精度可以用*来替代,在argi中指出就可以 —— printf ("%.*s",max,s); 格式: sprintf (char *string, char *format, arg1, arg2, ...) sprintf函数 作用:将format和argi的格式化后的结果存储到string中 格式: type func(argi, ...) — 确定参数写完后,在最后用...来处理可变数目参数 格式: va_list ap 数据类型: va_list 用来指向可变参数 格式: va_start (last_name_arg, ap) - 宏: va_start -用来对va_list进行赋值,使其指向可变数目参数中的第一个。 (也就是第一个非具名参数) / 使用stdargs.h中的数据、函数和宏来处理 格式: va_arg (ap, type) va_arg 函数对于可变数目的参数列表的处理方式 将ap指向的参数用type数据类型解析出来,并使va_list指向下一 格式: va_end (ap) /* minprintf: minimal printf with variable argument list */ void minprintf(char *fmt, ...) 从输入中根据format定义的数据类型将数据读取到参数中。参数 scanf函数 — 格式: int scanf(char *format, arg1, ...)-需要是pointer 将string中的数据通过format格式化到参数中。参数需要是 sscanf函数 — 格式: int sscanf(char *string, char *format, arg1, ...) pointer 头文件 stdio.h → 文件指针: FILE *fp name为文件名 w 写 文件打开函数: FILE *fopen(char *name, char *mode)_ input, output 文件打开 fp = fopen(name, mode) 读写模式 —— r 读 ■ a 添加 mode为打开方式 文本与二进制 —— b 二进制,不含b则为文本 文件关闭函数: int fclose(FILE *fp) int getc(FILE *fp) — 读取一个字符 int putc(int c, FILE *fp) — 写入一个字符,如果出错,则返回EOF int fscanf(FILE *fp, char *format, ...) - 文件访问 格式化操作 int fprintf(FILE *fp, char *format, ...) 文件操作 - int ferror(FILE *fp)—— 显式检查文件指针是否错误(可能因为磁盘掉了导致错误) 文件指针检查操作 int feof(FILE *fp) — 检查是否到文件尾 char *fgets(char *line, int maxline, FILE *fp)—— 读取一行,但是最多读取maxline-1个字符 行操作 int fputs(char *line, FILE *fp) — 放置一行,正常成功是返回0,出错时返回EOF stdin和键盘连接 C程序运行时OS会默认打开3个文件,standard input, - stdout, stderr和屏幕连接 standard output, standard error. getchar和putchar的宏: #define getchar() getc(stdin) #define putchar() putc((c), stdout) concatenate t to end of s strcat(s,t) concatenate n characters of t to end of s strncat(s,t,n) return negative, zero, or positive for strcmp(s,t) s < t, s == t, or s > tsame as stremp but only in first n characters strncmp(s,t,n) 字符串操作 —— string.h — strcpy(s,t) copy t to s copy at most n characters of t to s strncpy(s,t,n) return length of s strlen(s) return pointer to first c in s, or NULL if not present stichr(s,c) return pointer to last c in s, or NULL if not present strrchr(s,c) isalpha(c) non-zero if c is alphabetic, 0 if not isupper(c) non-zero if c is upper case, 0 if not islower(c) non-zero if c is lower case, 0 if not isdigit(c) non-zero if c is digit, 0 if not 🖊 字符测试和转换 —— ctype.h — isalnum(c) non-zero if isalpha(c) or isdigit(c), 0 if not isspace(c) non-zero if c is blank, tab, newline, return, formfeed, vertical tab toupper(c) return c converted to upper case tolower(c) return c converted to lower case 向文件流中放入一个字符,但是在该字符被读取前只能放入一 ✓ Ungetc — 格式: int ungetc(int c, FILE *fp) — 个,多次放入时,后放入的会覆盖先放入的。 / 命令行执行 — 格式: system(char *s) — 将s作为命令行参数进行执行 返回一个指针,该指针指向n个byte的存储空间的首地址,如果 void *malloc(size_t n) — 分配失败,则返回NULL。分配的存储空间未被初始化 其他库函数 返回一个指针,该指针指向n个数据的首地址,每个数据占据size 存储相关 — void *calloc(size_t n, size_t size) — 个byte,如果分配失败,则返回NULL。分配的存储空间被初始 化为0 free(p) — 释放由malloc或者calloc分配的指针 sine of x, x in radians sin(x)cosine of x, x in radians $\cos(x)$ atan2(y,x) arctangent of y/x, in radians exp(x)exponential function e^x natural (base e) logarithm of x (x>0)log(x)数学函数 —— math.h —— $\log 10(x)$ common (base 10) logarithm of x (x > 0) pow(x,y) x^y square root of $x (x \ge 0)$ sqrt(x)absolute value of xfabs(x)rand() — 返回一个伪随机int型数字 随机数 —— stdlib.h srand() — 为rand()设置seed · input/output —— shell有三个文件描述符: 0-stdin, 1-stdout, 2-stderr int n_read = read(int fd, char *buf, int n)
int n_written = write(int fd, char *buf, int n) fd为文件描述符,buf为待传数据或接收数据的缓冲区,n表示传 文件读写 输的字节数目 头文件 fcntl.h 在System V system上 sys/file.h 在BSD上 O_RDONLY - 文件打开 flags主要用到的几个值 — O_WRONLY int fd; O_RDWR int open(char *name, int flags, int perms) / file system — C在低级别进行文件调用的函数 perms — 在open函数中总是0 the UNIX system interface · 文件创建 —— int creat(char *name, int perms)—— perms —— 使用UNIX常用的权限来表示,例如:#define PERMS 0666 ·文件关闭 —— close(int fd) —— 对应于fclose,但是不同的是close不会进行缓冲区的冲洗。 删除一个文件名,如果该文件名是文件系统中最后一个指向该文件的文件名,则删除文件。否则只删除文件名 文件去关联 —— unlink(char *name) offset为相对于定位点的偏移 文件随机定位函数 —— long lseek(int fd, long offset, int origin)— **0**: 文件开头 origin决定定位点 — 1: 当前位置 2: 文件末尾 storage allocation appendix