进程终止

正常终止：1.从main返回;2.调用exit;3.调用\_exit或\_Exit;最后一个线程从其启动例程返回;5.从最后一个线程调用pthread\_exit;

异常终止：1.调用abort;2.接到一个信号;最后一个线程对取消请求作出响应;

return 的返回值给父进程传递信息;若没有return，main函数的返回值可能为0也可能为最后一条语句的返回值;

int atexit(void (\*function)(void));

钩子函数：exit时调用这些函数，顺序与登记时相反;

使用\_exit和\_Exit不调用钩子函数，也不执行IO清理;如果内存写越界可以调用，防止问题扩大;

命令行参数

int getopt(int argc, char \* const argv[], const char \*optstring);

optstring是要识别的字符所组成的字符串，函数返回值是一个字符转换成int类型，一次返回一个需要套一个循环;optstring最开始加-，当遇到不是字符串中字符的时候返回1，配合optind用于读入pathname;

extern char \*optarg;当optstring中的字符后面跟:时，识别到该字符后optarg会指向该字符后的修饰字符串，如-H 24中的24;

extern int optind会指向下一个要识别的位置;

int getopt\_long(int argc, char \* const argv[], const char \*optstring, const struct option \*longopts, int \*longindex);

环境表

extern char \*\*environ;类似与argv是一个字符串数组，里面每个字符串代表一个环境变量;用于遍历所有环境变量;

char \*getenv(const char \*name);

返回指向name=value中value的字符串，失败返回NULL;name则是name=value中name的字符串;

int setenv(const char \*name, const char \*value, int overwrite);

int unsetenv(const char \*name);

成功返回0,失败返回-1;overwrite为0则若原环境变量存在就不修改，非零则覆盖;

int setjmp(jmp\_buf env);

直接调用返回0,从longjmp返回则非0;

void longjmp(jmp\_buf env, int val);

快速返回指定位置，env必须是全局变量，通常用于多层嵌套的“const函数”;

int getrlimit(int resource, struct rlimit \*rlim);

int setrlimit(int resource, const struct rlimit \*rlim);

普通进程可以升降软限制，只能降低硬限制;root进程可以升降软硬限制;