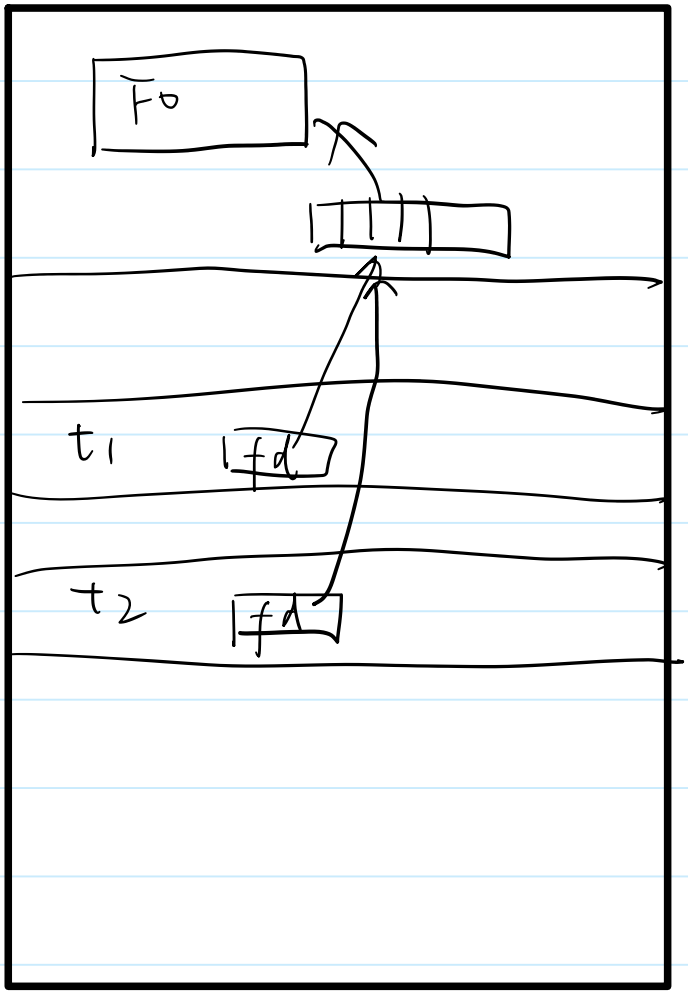
文件对象和多线程

2023年8月16日 9:31



不是dup，只用了一个fd.

```c
void * threadFunc(void *arg){
    int *pfd = (int *)arg;
    int fd = *pfd;
    sleep(1);
    ssize_t sret = write(fd,"hello",5);
    printf("sret = %ld\n", sret);
    ERROR_CHECK(sret,-1,"write");
    pthread_exit(NULL);
}
int main(int argc, char *argv[])
{
    int fd = open("file1",O_RDWR);
    pthread_t tid;
    pthread_create(&tid,NULL,threadFunc,&fd);
    close(fd);
    pthread_join(tid,NULL);
    return 0;
}
```

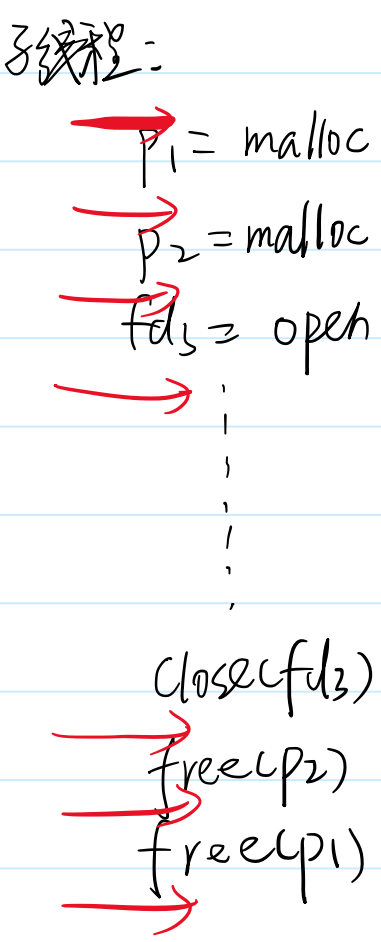
使用堆空间

2023年8月16日 11:13

```c
void * threadFunc(void *arg){
    int * p = (int *)arg;
    sleep(1);
    ++*p;
    printf("child, *p = %d\n", *p);
}
int main(int argc, char *argv[])
{
    int * p = (int *)malloc(sizeof(int));
    *p = 1000;
    pthread_t tid;
    pthread_create(&tid,NULL,threadFunc,p);
    //free(p);
    pthread_join(tid,NULL);
    free(p);
    return 0;
}
```

资源清理

2023年8月16日 11:14



资源清理栈

① 元素 函数指针 + 参数

② LIFO.

压栈和弹栈

2023年8月16日 11:19

```
void pthread_cleanup_push(void (*routine)(void *),
                          void *arg);
void pthread_cleanup_pop(int execute);
```

压 ← 清理函数. ← 参数

弹.

问题：不同类型的函数指针如何封装？

```
void (*p1)(int)
void (*p2)(double)
```

用好 双胞

```
void p3(int){
    p2(...);
}
```

压栈和弹栈

2023年8月16日 11:33

要求① 申请资源之后，马上压栈。

② 把原来的清理函数 替换成弹栈。

```
void *threadFunc(void *arg){
    void * p1 = malloc(1024);
    void * p2 = malloc(1024);
    int fd3 = open("file1",O_RDWR);

    closefd3(&fd3);
    freep2(p2);
    freep1(p1);
}
```

```
void *threadFunc(void *arg){
    void * p1 = malloc(1024);
    pthread_cleanup_push(freep1,p1);
    void * p2 = malloc(1024);
    pthread_cleanup_push(freep2,p2);
    int fd3 = open("file1",O_RDWR);
    pthread_cleanup_push(closefd3,&fd3);

    //closefd3(&fd3);
    pthread_cleanup_pop(1);
    //freep2(p2);
    pthread_cleanup_pop(1);
    //freep1(p1);
    pthread_cleanup_pop(1);
}
```

压栈和弹栈的原理

2023年8月16日 11:40

push 压栈.

当线程 被cancel 且处理到取消点 时，如果栈非空.
or pthread_exit

就会才栈全部弹出来并调用.

自动调用 pthread_cleanup_pop、且参数是0.

弹出一个并调用

申请资源马上压栈，压栈是一个取消点

注意的地方

2023年8月16日 11:51

① 使用return 终止线程 不会释放资源清理.

② 只push 不pop 解决编译错误

```
gcc 6_cleanup.c -o 6_cleanup -g -pthread
6_cleanup.c: In function ‘threadFunc’:
6_cleanup.c:32:1: error: expected ‘while’ before ‘int’
   32 | int main(int argc, char *argv[])
      | ^~~
6_cleanup.c:38:1: error: expected declaration or statement at end of in
put
   38 | }
      | ^
```

```
# define pthread_cleanup_push(routine, arg) \
  do {                                      \
    __pthread_cleanup_class __clframe (routine, arg)
```

do{ }

x~~~次

```
/* Remove a cleanup handler installed by the matching pthread_cleanup_push.
   If EXECUTE is non-zero, the handler function is called. */
# define pthread_cleanup_pop(execute) \
    __clframe.__setdoit (execute);    \
  } while (0)
```

0 不释放

>0

push和pop成对出现，同一个作用域

push和pop成对出现，同一个作用域