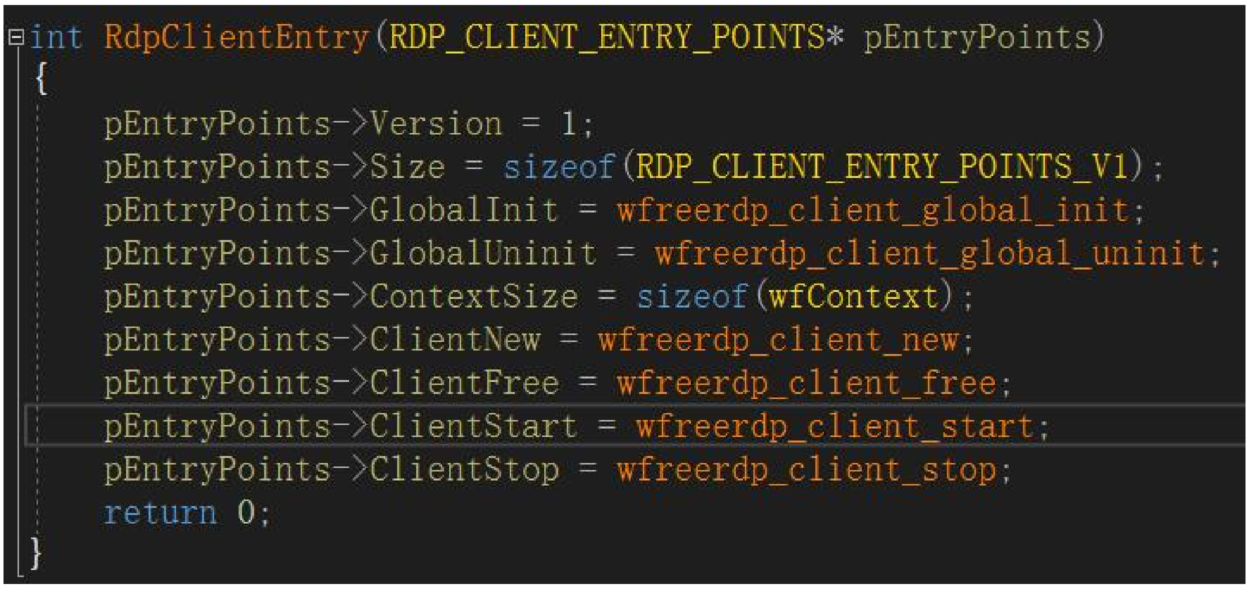
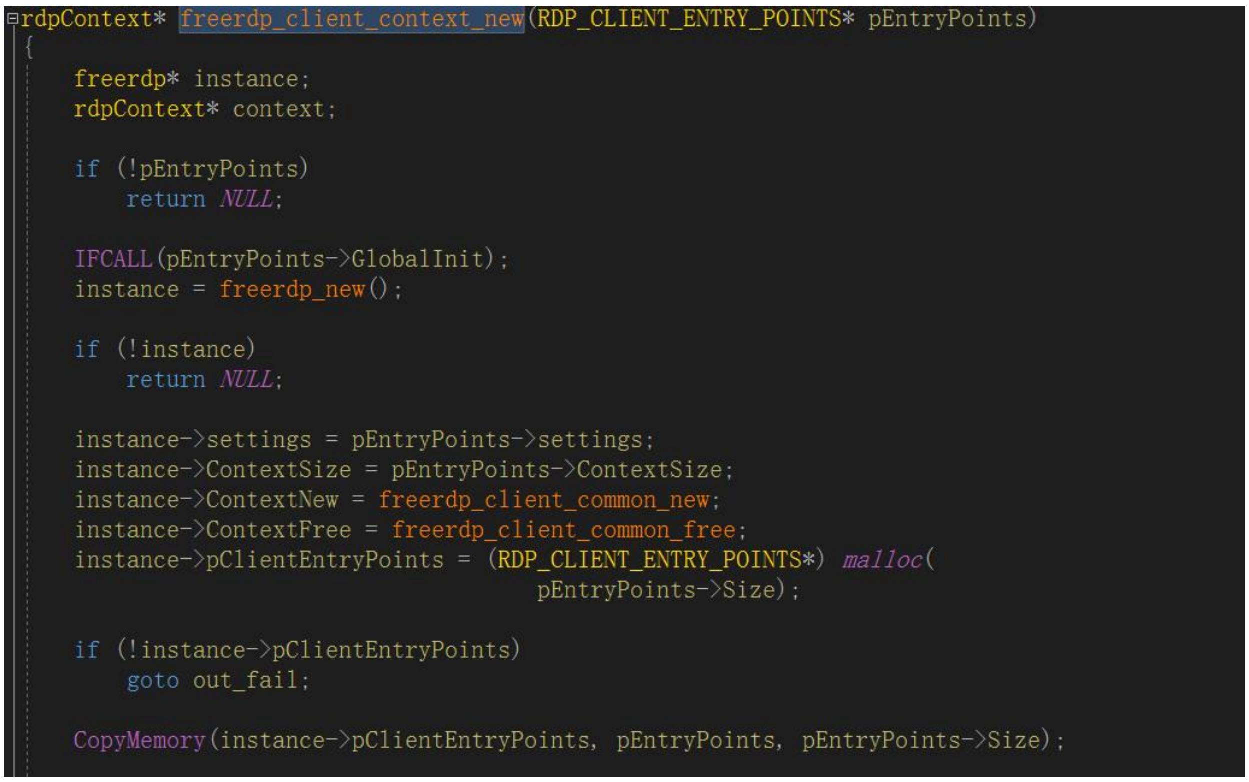
1. RdpClientEntry



```
int RdpClientEntry(RDP_CLIENT_ENTRY_POINTS* pEntryPoints)
{
	pEntryPoints->Version = 1;
	pEntryPoints->Size = sizeof(RDP_CLIENT_ENTRY_POINTS_V1);
	pEntryPoints->GlobalInit = wfreerdp_client_global_init;
	pEntryPoints->GlobalUninit = wfreerdp_client_global_uninit;
	pEntryPoints->ContextSize = sizeof(wfContext);
	pEntryPoints->ClientNew = wfreerdp_client_new;
	pEntryPoints->ClientFree = wfreerdp_client_free;
	pEntryPoints->ClientStart = wfreerdp_client_start;
	pEntryPoints->ClientStop = wfreerdp_client_stop;
	return 0;
}
```

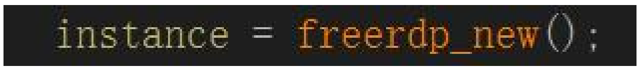
- 为pEntryPoints的各个成员赋值。 其中,wfreerdp_client_global_init主要是为GlobalInit绑定wfreerdp_client_global_init, 该接口主要为freerdp_load_static_channel_addin_entry绑定freerdp_channels_load_static_addin_entry()接口。freerdp_channels_load_static_addin_entry是将事先准备好的. 此接口最终是为wf_pre_connect中的freerdp_client_load_addins服务的
- 为ClientNew绑定wfreerdp_client_new此接口中主要为instance绑定了wf_pre_connect和wf_post_connect接口, 还有输入账号密码的回调。稍后介绍post和pre。
- 为ClientStart绑定启动客户端的回调wfreerdp_client_start

2. freerdp_client_context_new



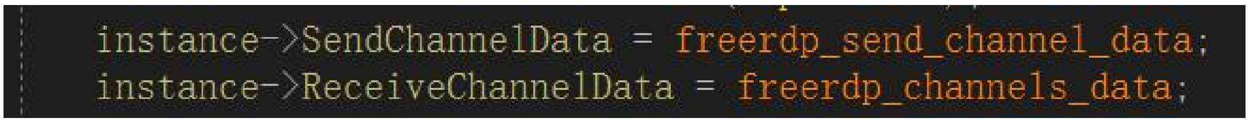
```
rdpContext* freerdp_client_context_new(RDP_CLIENT_ENTRY_POINTS* pEntryPoints)
{
	freerdp* instance;
	rdpContext* context;

	if (!pEntryPoints)
		return NULL;

	IFCALL(pEntryPoints->GlobalInit);
	instance = freerdp_new();

	if (!instance)
		return NULL;

	instance->settings = pEntryPoints->settings;
	instance->ContextSize = pEntryPoints->ContextSize;
	instance->ContextNew = freerdp_client_common_new;
	instance->ContextFree = freerdp_client_common_free;
	instance->pClientEntryPoints = (RDP_CLIENT_ENTRY_POINTS*) malloc(
	                                   pEntryPoints->Size);

	if (!instance->pClientEntryPoints)
		goto out_fail;

	CopyMemory(instance->pClientEntryPoints, pEntryPoints, pEntryPoints->Size);
```

- IFCALL(pEntryPoints->GlobalInit)主要执行1.中 绑定的接口, 详细请看1.



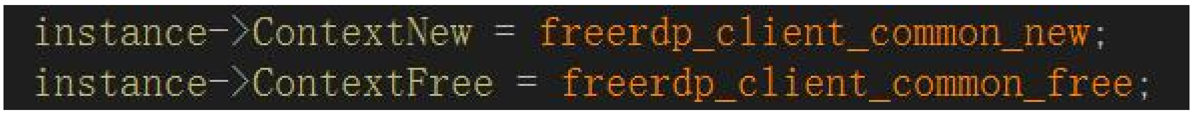
```
instance = freerdp_new();
```

- 创建instance对象主要为instance绑定send和receive回调



```
instance->SendChannelData = freerdp_send_channel_data;
instance->ReceiveChannelData = freerdp_channels_data;
```

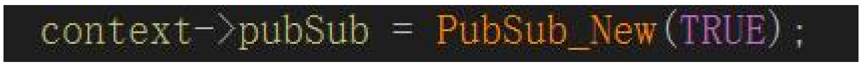
- 为contexnew和contextfree绑定回调, 这两个回调主要调用了1.中绑定的wfreerdp_client_new和wfreerdp_client_free



```
instance->ContextNew = freerdp_client_common_new;
instance->ContextFree = freerdp_client_common_free;
```

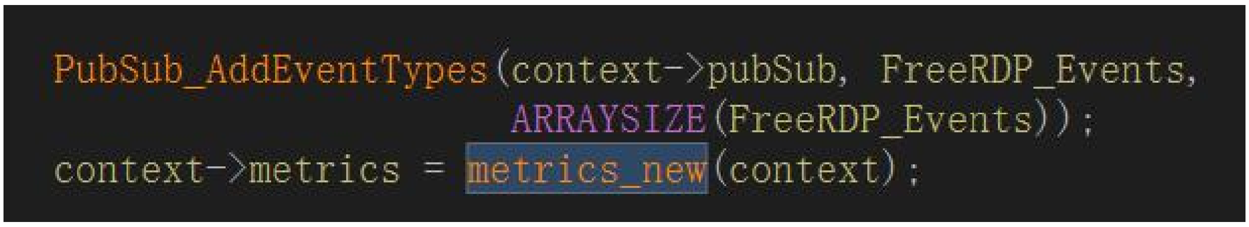
3. freerdp_context_new

- 目前不知道做什么的



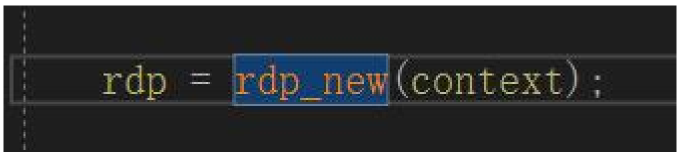
```
context->pubSub = PubSub_New(TRUE);
```

- 同上



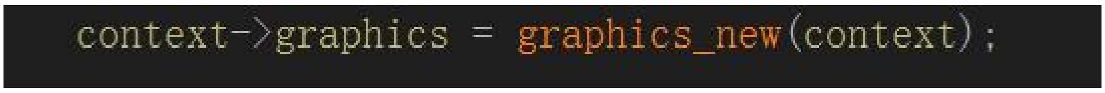
```
PubSub_AddEventTypes(context->pubSub, FreeRDP_Events,
                     ARRAYSIZE(FreeRDP_Events));
context->metrics = metrics_new(context);
```

- 创建rdp对象, 如果setting没有初始化过则初始化setting, 创建transport license input update等对象。仅仅是创建, 即开辟改对象大小的内存, 没有做其他的操作, 目前是这样。



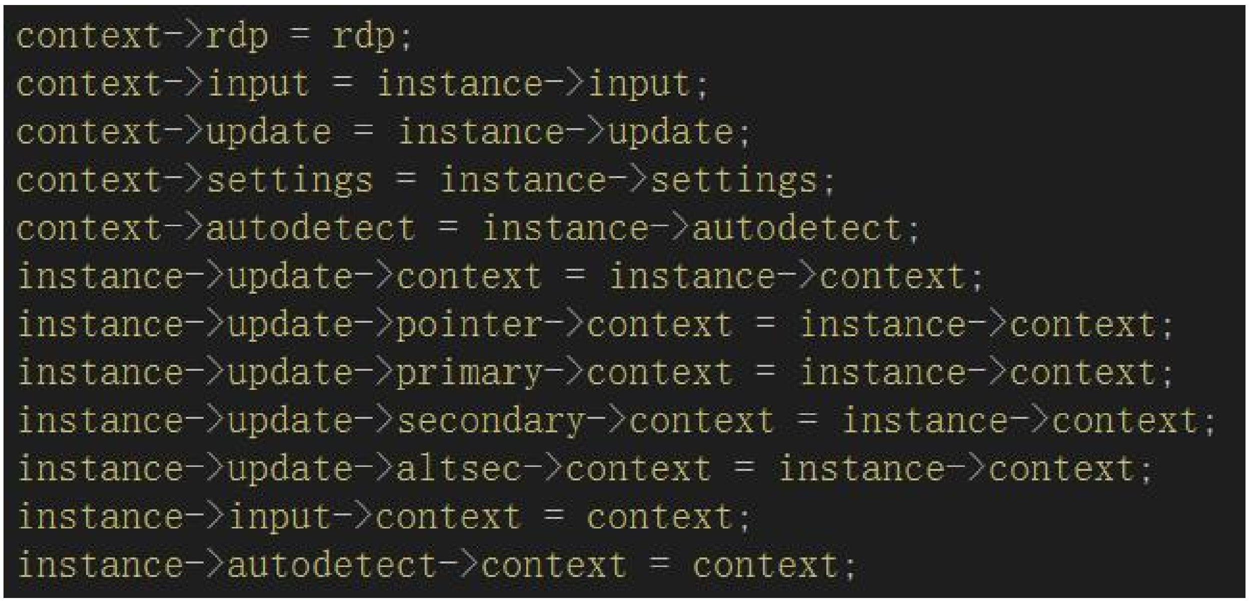
```
    rdp = rdp_new(context);
```

- 创建图形对象, 只是为成员开辟内存以及为各个成员的new成员绑定接口, 其他的接口在*_register_*类型的接口中绑定, 这些接口实在wf_post_connect()函数中执行, 也就是成功链接FreeRDP服务器之后执行。



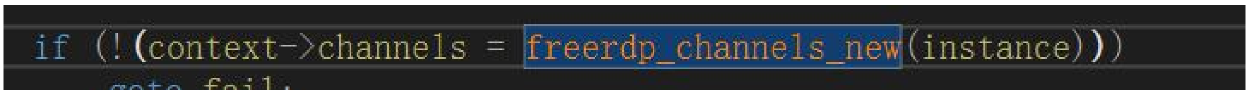
```
  context->graphics = graphics_new(context);
```

- 交换context和instance数据



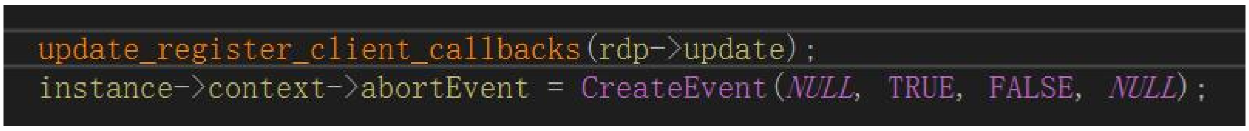
```
context->rdp = rdp;
context->input = instance->input;
context->update = instance->update;
context->settings = instance->settings;
context->autodetect = instance->autodetect;
instance->update->context = instance->context;
instance->update->pointer->context = instance->context;
instance->update->primary->context = instance->context;
instance->update->secondary->context = instance->context;
instance->update->altsec->context = instance->context;
instance->input->context = context;
instance->autodetect->context = context;
```

- 作用目前不清楚



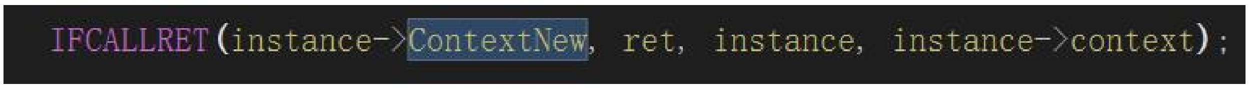
```
 if (!(context->channels = freerdp_channels_new(instance)))
```

- 创建channels对象,主要是消息队列的创建, 哈希表的创建, 还有锁, 目前哈希表的作用还不清楚。



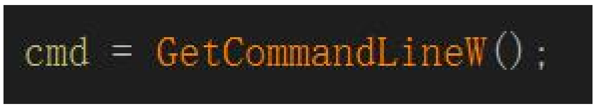
```
 update_register_client_callbacks(rdp->update);
 instance->context->abortEvent = CreateEvent(NULL, TRUE, FALSE, NULL);
```

- 最终调用此接口, 简介调用wfreerdp_client_new, 这里面是绑定pre post start接口。



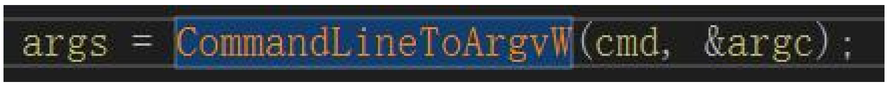
```
  IFCALLRET(instance->ContextNew, ret, instance, instance->context);
```

4.获取命令行输入



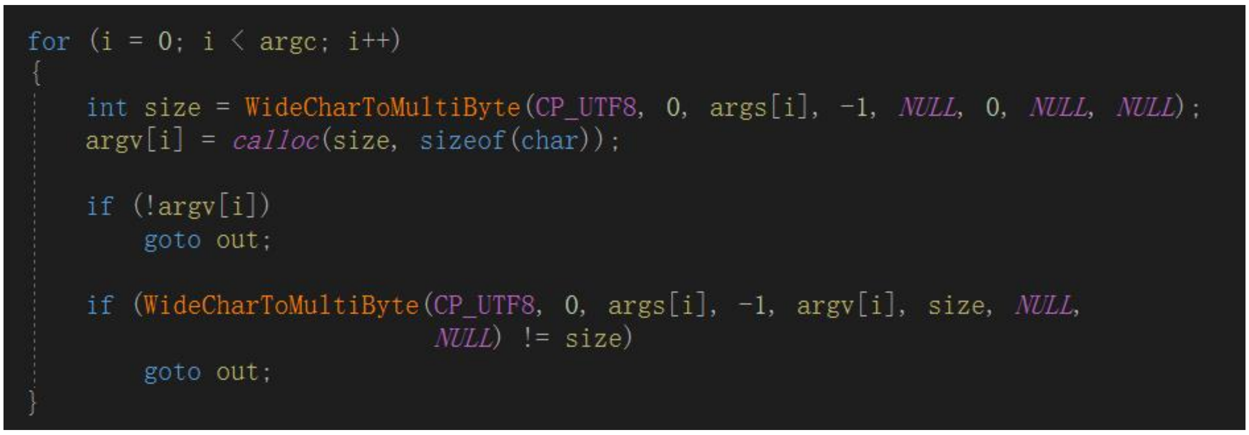
```
 cmd = GetCommandLineW();
```

5. 将命令行输入解析成命令列表



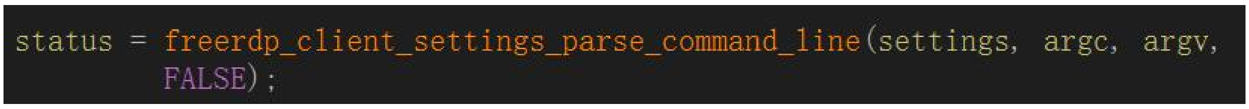
```
 args = CommandLineToArgvW(cmd, &argc);
```

6.参数字符转化为宽字符



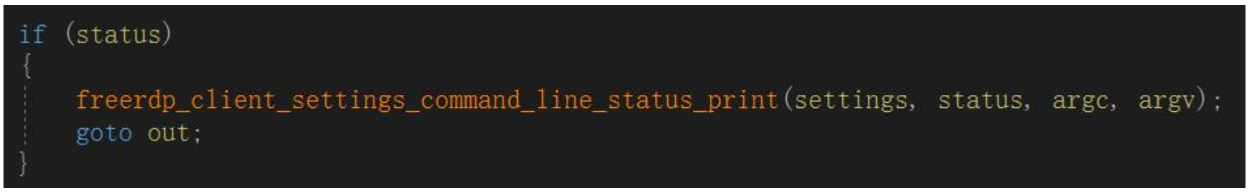
```
for (i = 0; i < argc; i++)
{
    int size = WideCharToMultiByte(CP_UTF8, 0, args[i], -1, NULL, 0, NULL, NULL);
    argv[i] = calloc(size, sizeof(char));

    if (!argv[i])
        goto out;

    if (WideCharToMultiByte(CP_UTF8, 0, args[i], -1, argv[i], size, NULL,
                            NULL) != size)
        goto out;
}
```

7.将解析的命令行参数, 赋值给setting



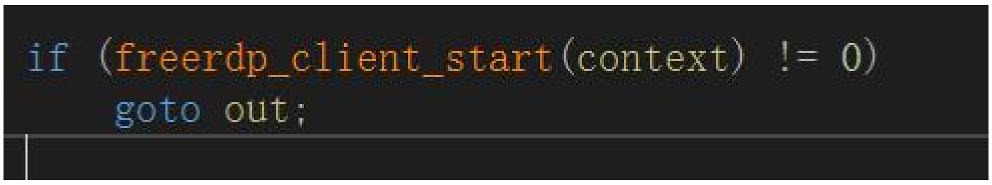
```
status = freerdp_client_settings_parse_command_line(settings, argc, argv,
         FALSE);
```

8. 如果失败, 打印帮助



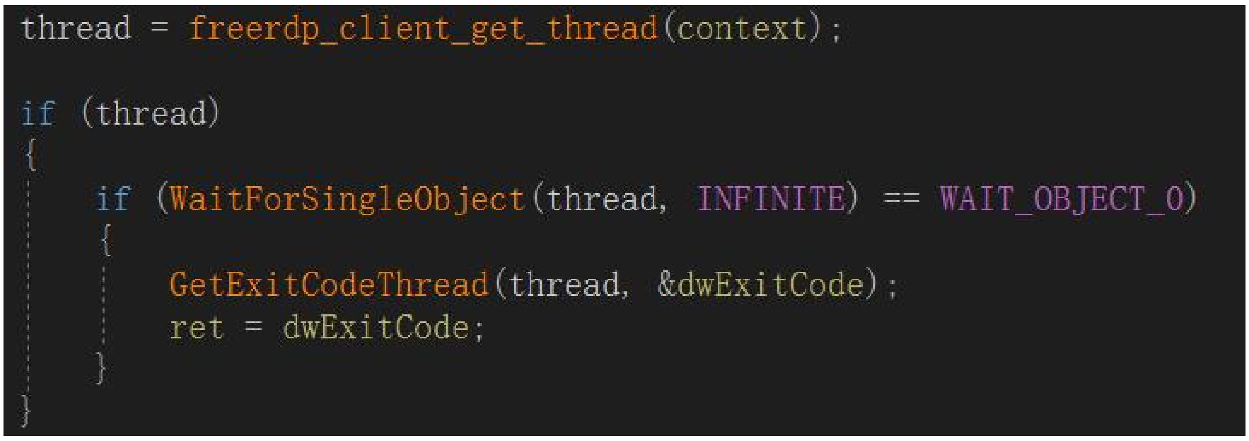
```
if (status)
{
    freerdp_client_settings_command_line_status_print(settings, status, argc, argv);
    goto out;
}
```

10. 正式启动客户端, 内部简介调用了wfreerdp_client_start此接口在RdpClientEntry中绑定。



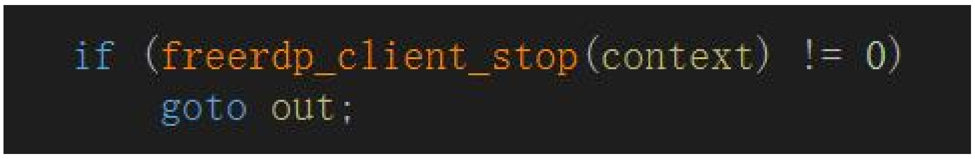
```
if (freerdp_client_start(context) != 0)
    goto out;
```

11.获取客户端线程, 并等待线程结束



```
thread = freerdp_client_get_thread(context);

if (thread)
{
    if (WaitForSingleObject(thread, INFINITE) == WAIT_OBJECT_0)
    {
        GetExitCodeThread(thread, &dwExitCode);
        ret = dwExitCode;
    }
}
```

12. 完成资源回收工作, 简介调用wfreerdp_client_stop接口。



```
if (freerdp_client_stop(context) != 0)
    goto out;
```