SWC 虚拟项目

测试报告

版本号: 0.1

发布日期: 2020-09-24

版本历史

版本号	日期	制/修订人	内容描述
0.1	2020.9.16	汤健雄	1. 建立文档

目　录

表　格

1 前言

1.1 文档简介

此文档为 P166 项目软件 demo 的测试报告。

1.2 目标读者

P166 项目开发者和验收人员。

1.3 项目背景

M 公司是一家采用全志 SOC 集成方案的品牌大客户,7 月份跟全志合作立项, 远程联合开发一个代号 P166 的重要项目, 该项目基于全志 A100 平台, 为了加快项目并行进度,P166 项目客户端项目经理 L, 向全志 Aserver 平台提交了一项软件开发需求, 要求全志方提供一套易用、稳定、可复用的软件 Demo, 降低客户端前期开发工作量, 加快二次开发整体进度。

- 项目负责人：苏佳佳
- 参与人员：汤健雄、FAE 主管、项目经理

1.3.1 功能需求

软件 Demo 包含后台服务应用 A、客户端应用 B、客户端应用 C 和内核模块 K 四个独立组件。K 作为 A 和 B、A 和 C 之间的通信中转站,B 和 C 之间不能通信。

Demo 功能 1：A 和 B 发生一次通信,A 将数据包编码后发送给 K,K 受到数据包转发给 B,B 对数据包完成逆向解码还原, 并将原始数据的 HASH 值字符串通过 K 返还给 A。A 受到 HASH 值字符串进行正确性校验, 校验成功完成通信, 校验失败后 Log 日志抛出异常码 ERN110。

Demo 功能 2：同理 A 和 C 发生通信过程如上, 校验失败后 Log 日志抛出异常码 ERN120。

1.4 限制条件

- 规格软件开发：保证解耦设计，可被二次定制，具有一定的鲁棒性
  代码规范：代码风格符合 SWC 和 SW4 的代码规范要求，使用 git 进行统一的管理
  测试：各个模块支持多种方便、单独的调试手段，支持临时数据的调试，支持命令调试
  文档：符合软件设计文档规范，并需在内部评审通过
- 交付说明代码：提交至 git 仓库——SWC-Bootcamp

文档：上传至 edoc，具体文档包括：虚拟项目任务计划书，软件概要设计文档，各个组件的测试列表、测试报告，各个模块代码的静态代码检查报告，组件之间的联调报告，代码的 ROM/RAM 分析报告，开发、调试过程的记录文档，总结文档。

1.5 测试目标

本次测试是针对 P166 项目进行的验收测试，目的为了判定改系统是否满足交付要求中规定的功能与性能指标。

1.6 测试范围及方法

表 1-1：测试范围及方法

序号	测试项目	测试方法	测试工具
1	安装部署测试	黑盒/手工	无
2	模块功能测试	黑盒/手工	shell 脚本
3	系统功能测试	黑盒/手工	shell 脚本
4	联调测试	黑盒/手工	shell 脚本
5	性能测试	黑盒/手工	无
6	安全性	黑盒/手工	无

1.7 测试环境

本软件 demo 的测试运行环境主要分为三个部分,Windows pc、Linux 编译器和目标硬件板。主要过程是代码通过 Linux 编译器编译成功之后，在 Windows pc 上通过串口工具和其他调试工具将编译产物推送到目标硬件板上进行测试。

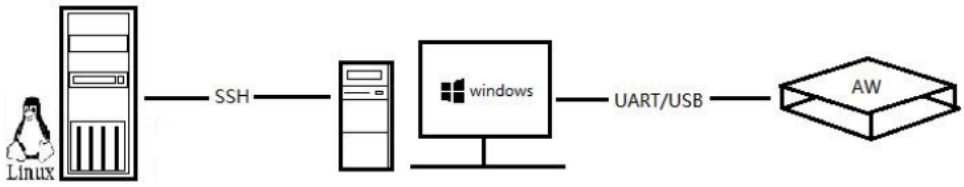

图 1-1: ceshitupian

表 1-2: 总体运行环境

测试环境	硬件配置	系统版本	网络环境	依赖工具
Linux 编译服务器	硬盘 30G, 支持 64 位系统	Ubuntu 14.04.5 LTS	正常网络环境	gcc,ncurse,bison,autoconf, wget,patch,texinfo 和 zlib 等
Windows PC	正常配置	Windows 7	正常网络环境	串口工具和 adb 工具
硬件开发板	T507 开发板	Linux kunos 4.9.170	正常网络环境	shell 脚本、GDB 工具

2 安装部署测试

2.1 编译运行测试

测试源码是否能正常编译部署运行。

2.1.1 前置条件

- Linux 编译服务器具备相关的交叉编译工具, 且环境变量设置正确
- Windows PC 与开发板连接成功, 串口工具与 adb 工具运行正常
- 开发板系统运行正常

2.1.2 测试方法

- 下载相关源码, 进入顶层目录
- 直接执行编译脚本：./script/build.sh
- 采用 adb push 工具将 out 目录下的产物推送到开发板环境中
- 串口或 adb 环境运行所有的编译产物

2.1.3 测试通过条件

- 编译无警告和错误
- out 目录下生成：client_b,client_c,server_a 和 kernel_k.ko 四项编译产物
- 将编译产物推送到开发板环境可正常执行

2.1.4 测试结论

编译无警告, 产物生成正常, 推送到开发板上可正常运行, 安装部署测试通过。

3 功能测试

3.1 编码功能测试

测试编解码模块编码功能

3.1.1 前置条件

- test/encode_test/目录下代码编译成功无告警

3.1.2 测试用例

表 3-1: 编码功能测试用例

测试用例	内容描述	备注	通过条件	通过情况
encode_test_1	NULL 空指针	异常测试	异常返回值-2	pass
encode_test_2	字符串长度 1+	正常测试	编码成功	pass
encode_test_3	字符串长度 10+	正常测试	编码成功	pass
encode_test_4	字符串长度 100+	正常测试	编码成功	pass
encode_test_5	字符串长度 200+	正常测试	编码成功	pass
encode_test_6	字符串长度 500+	正常测试	编码成功	pass
encode_test_7	字符串长度 800+	正常测试	编码成功	pass
encode_test_8	字符串长度 1400	边界值测试	编码成功	pass
encode_test_9	字符串长度 1401+	异常测试	返回异常值-3	pass

3.1.3 测试方法

- 进入 test/encode_test/, 执行 make 命令
- 根据测试用例, 运行测试文件, 查看测试用例通过情况

3.1.4 测试通过条件

- 通过所有的测试用例

3.1.5 测试结论

编码功能测试通过

3.2 解码功能测试

测试编解码模块的解码功能

3.2.1 前置条件

- test/decode_test/目录下代码编译成功无告警

3.2.2 测试用例

表 3-2: 解码功能测试用例

测试用例	内容描述	备注	通过条件	通过情况
decode_test_1	NULL 空指针	异常测试	异常返回值-2	pass
decode_test_2	字符串长度 1+	正常测试	解码成功	pass
decode_test_3	字符串长度 10+	正常测试	解码成功	pass
decode_test_4	字符串长度 100+	正常测试	解码成功	pass
decode_test_5	字符串长度 200+	正常测试	解码成功	pass
decode_test_6	字符串长度 500+	正常测试	解码成功	pass
decode_test_7	字符串长度 800+	正常测试	解码成功	pass
decode_test_8	字符串长度 1400	边界值测试	解码成功	pass
decode_test_9	字符串长度 1401+	异常测试	返回异常值-3	pass

3.2.3 测试方法

- 进入 test/decode_test/, 执行 make 命令, 生成测试文件
- 根据测试用例, 运行测试文件, 查看测试用例通过情况

3.2.4 测试通过条件

- 通过所有的测试用例

3.2.5 测试结论

解码功能测试通过

3.3 字符串 hash 计算测试

测试 hash 模块的字符串 hash 值计算功能

3.3.1 前置条件

- test/hash_str_test/目录下代码编译成功无告警

3.3.2 测试用例

表 3-3: 字符串 hash 值计算功能测试用例

测试用例	内容描述	备注	通过条件	通过情况
hashstr_test_1	NULL 空指针	异常测试	异常返回值-2	pass
hashstr_test_2	字符串长度 1+	正常测试	hash 值正确计算	pass
hashstr_test_3	字符串长度 10+	正常测试	hash 值正确计算	pass
hashstr_test_4	字符串长度 100+	正常测试	hash 值正确计算	pass
hashstr_test_5	字符串长度 200+	正常测试	hash 值正确计算	pass
hashstr_test_6	字符串长度 500+	正常测试	hash 值正确计算	pass
hashstr_test_7	字符串长度 800+	正常测试	hash 值正确计算	pass
hashstr_test_8	字符串长度 1400	边界值测试	hash 值正确计算	pass
hashstr_test_9	字符串长度 1401+	异常测试	返回异常值-3	pass

3.3.3 测试方法

- 进入 test/hash_str_test/, 执行 make 命令
- 根据测试用例, 运行测试文件, 查看测试用例通过情况

3.3.4 测试通过条件

- 通过所有的测试用例

3.3.5 测试结论

字符串 hash 值计算功能测试通过

3.4 文件 hash 值计算测试

测试 hash 模块的文件 hash 值计算功能

3.4.1 前置条件

- test/hash_file_test/目录下代码编译成功无告警

3.4.2 测试用例

测试用例	内容描述	备注	通过条件	通过情况
hashfile_test_1	NULL	空指针	异常返回值-2	pass
hashfile_test_2	tysiadasd	不存在的相对路径文件	异常返回值-1	pass

3.5 文件 hash 值计算测试

测试 hash 模块的文件 hash 值计算功能

3.5.1 前置条件

- test/hash_file_test/目录下代码编译成功无告警

3.5.2 测试用例

表 3-5: 文件 hash 值计算测试用例

测试用例	内容描述	备注	通过条件	通过情况
hashfile_test_1	NULL	空指针	异常返回值-2	pass
hashfile_test_2	tysiadasd	不存在的相对路径文件	异常返回值-1 打印异常信息	pass
hashfile_test_3	/tysi/adasd	不存在的绝对路径文件	hash 值正确计算	pass
hashfile_test_4	Makefile	相对路径文件	hash 值正确计算	pass
hashfile_test_5	1.txt	相对路径文本文件	hash 值正确计算	pass
hashfile_test_6	1.jpg	相对路径图片文件	hash 值正确计算	pass
hashfile_test_7	/1.txt	绝对路径文件	hash 值正确计算	pass

3.5.3 测试方法

- 进入 test/hash_file_test/, 执行 make 命令
- 根据测试用例, 运行测试文件, 查看测试用例通过情况

3.5.4 测试通过条件

- 通过所有的测试用例

3.5.5 测试结论

文件 hash 值计算功能测试通过

3.6 封装功能测试

测试封装模块的功能是否正常

3.6.1 前置条件

- test/pack_test/目录下代码编译成功无告警

3.6.2 测试用例

测试用例	内容描述	备注	通过条件	通过情况
pack_test_1	NULL,a,b,m	空字符指针, 其他内容正常	异常返回值-2	pass
pack_test_2	字符串（长度 123）,a,b,q	未知消息类型	异常打印	pass
pack_test_3	字符串（长度 200+）,a,b,m	正常输入	正确封装	pass
pack_test_4	字符串（长度 500+）,a,b,m	正常输入	正确封装	pass

3.7 封装功能测试

测试封装模块的功能是否正常

3.7.1 前置条件

- test/pack_test/目录下代码编译成功无告警

3.7.2 测试用例

表 3-7: 封装功能测试用例

测试用例	内容描述	备注	通过条件	通过情况
pack_test_1	NULL,a,b,m	空字符指针, 其他内容正常	异常返回值-2	pass
pack_test_2	字符串（长度 123）,a,b,q	未知消息类型	异常打印	pass
pack_test_3	字符串（长度 200+）,a,b,m	正常输入	正确封装	pass
pack_test_4	字符串（长度 500+）,a,b,m	正常输入	正确封装	pass
pack_test_5	字符串（长度 800+）,a,b,m	正常输入	正确封装	pass
pack_test_6	字符串（长度 1000+）,a,b,m	正常输入	正确封装	pass
pack_test_7	随机字符串（长度 1301+）,a,b,m	字符串长度超过最大值	返回异常值-3	pass

3.7.3 测试方法

- 进入 test/pack_test/, 执行 make 命令
- 根据测试用例, 运行测试文件, 查看测试用例通过情况

3.7.4 测试通过条件

- 通过所有的测试用例

3.7.5 测试结论

封装功能测试通过

3.8 解封装功能测试

测试解封装模块的功能是否正常

3.8.1 前置条件

- test/unpack_test/目录下代码编译成功无告警

3.8.2 测试用例

测试用例	内容描述	备注	通过条件	通过情况
unpack_test_1	NULL	空字符指针, 其他内容正常	异常返回值-2	pass
unpack_test_2	字符串（长度 10+）	正常输入	输出正确消息发送者和消息类型	pass
unpack_test_3	封装字符串（长度 200+）	正常输入	输出正确消息	pass

3.9 解封装功能测试

测试解封装模块的功能是否正常

3.9.1 前置条件

- test/unpack_test/目录下代码编译成功无告警

3.9.2 测试用例

表 3-9: 封装功能测试用例

测试用例	内容描述	备注	通过条件	通过情况
unpack_test_1	NULL	空字符指针, 其他内容正常	异常返回值-2	pass
unpack_test_2	字符串（长度 10+）	正常输入	输出正确消息发送者和消息类型	pass
unpack_test_3	封装字符串（长度 200+）	正常输入	输出正确消息发送者和消息类型	pass
unpack_test_4	封装字符串（长度 500+）	正常输入	输出正确消息发送者和消息类型	pass
unpack_test_5	封装字符串（长度 800+）	正常输入	输出正确消息发送者和消息类型	pass
unpack_test_6	封装字符串（长度 1000+）	正常输入	输出正确消息发送者和消息类型	pass
unpack_test_7	封装字符串（长度 1400+）	字符串长度超过最大值	返回异常值-3	pass

3.9.3 测试方法

- 进入 test/unpack_test/, 执行 make 命令
- 根据测试用例, 运行测试文件, 查看测试用例通过情况

3.9.4 测试通过条件

- 通过所有的测试用例

3.9.5 测试结论

封装功能测试通过

3.10 netlink 初始化测试

测试 netlink 模块的 netlink 初始化功能是否正常, 主要包括用户态 netlink socket 创建、地址初始化和 bind 地址绑定等功能是否成功。

3.10.1 前置条件

- 进入 test/netlink_test/测试用例编译无告警
- 生成产物可在目标开发板上正常运行，kernel_test.ko 模块成功运行

3.10.2 测试用例

表 3-10: netlink 初始化测试用例

测试用例	内容描述	备注	通过条件	通过情况
netlink_test_1	内核模块未加载 直接初始化用户态 netlink	异常测试	打印创建 套接字失败 [errno]93	pass
netlink_test_2	加载内核模块 使用同一个端口号初始化两次	异常测试	打印 bind 失败 ，地址已被使用	pass
netlink_test_3	加载内核模块 执行 netlink 初始化	正常测试	打印初始化成功 ，返回正确 socket 描述符	pass

3.10.3 测试方法

- 进入 test/netlink_test/，执行./build.sh 编译测试用例
- 将生成的 out/目录下的产物推送到目标开发板上，增加执行权限
- 按照测试用例的描述进行测试，观测相应的结果

3.10.4 测试通过条件

通过所有的测试用例，除了能够支持正常的用户态初始化功能之外，还需要对一些异常情况作出相应的反馈。

3.10.5 测试结论

netlink 初始化功能正常，且能够对一些异常情况作出相应的处理。

3.11 netlink 模块消息功能测试

测试 netlink 模块的消息发送与接收功能是否正常，主要体现在能够调用相关接口函数与内核模块进行正常通信。

3.11.1 前置条件

- 进入 test/netlink_test/测试用例编译无告警

- 生成产物可在目标开发板上正常运行，kernel_test.ko 模块成功运行

3.11.2 测试用例

表 3-11：netlink 消息发送接收测试用例

测试用例	内容描述	备注	通过条件	通过情况
netlink_sendrecv_test_1	发送长度为 1400 以内的随机字符串	正常测试	内核收到字符串并转发，用户态成功接收	pass
netlink_sendrecv_test_2	快速循环发送长度为 1400 的随机字符串	高频次测试	每次发送和接收都正常	pass
netlink_sendrecv_test_3	当消息发送时移除内核模块	异常测试	发送失败，打印异常码	pass

3.11.3 测试方法

- 进入 test/netlink_test/，执行./build.sh 编译测试用例
- 将生成的 out/目录下的产物推送到目标开发板上，增加执行权限
- 按照测试用例的描述进行测试，观测相应的结果

3.11.4 测试通过条件

通过所有的测试用例，除了能够支持正常的用户态内核态通信之外，还需要完成大量快速的通信的收发和一些异常情况的处理。

3.11.5 测试结论

netlink 消息功能正常，可与内核之间完成大量数据的快速收发，也可以对一些异常情况作出处理。

3.12 系统消息通信功能测试

验证系统的消息通信功能是否正常，首先保证 Demo 功能正常，即 AB 通信、AC 通信正常。然后保证多次通信数据传输过程的每个环节都正常工作。最后，对可能出现的一些异常情况进行测试。

3.12.1 前置条件

- 源码编译无告警，能正常生成产物，能在目标板上正常运行
- 各模块 netlink 初始化功能正常
- 各公共模块功能正常

3.12.2 测试用例

表 3-12: 系统通信功能测试

测试用例	内容描述	备注	通过条件	通过情况
netlink_msg_test_1	b 向 a 发送长度为 1400 以内的随机字符串	正常测试	各环节工作正常	pass
netlink_msg_test_2	a 向 b 发送长度为 1400 以内的随机字符串	正常测试	各环节工作正常	pass
netlink_msg_test_3	a 向 c 发送长度为 1400 以内的随机字符串	正常测试	各环节工作正常	pass
netlink_msg_test_4	c 向 a 发送长度为 1400 以内的随机字符串	正常测试	各环节工作正常	pass
netlink_msg_test_5	ab 高频次互发长度为 1400 以内的随机字符串	正常测试	各环节工作正常	pass
netlink_msg_test_6	ac 高频次互发长度为 1400 以内的随机字符串	正常测试	各环节工作正常	pass
netlink_msg_test_7	发送长度为 1400+ 的字符串	异常测试	发送端提示长度超过最大长度无法发送	pass
netlink_msg_test_8	bc 之间发送消息	异常测试	内核丢弃消息并向发送方反馈非法通信	pass
netlink_msg_test_9	通信过程中接收端进程被中止	异常测试	内核向反馈消息接收端未运行	pass
netlink_msg_test_10	b、c 同时向 a 发送长度不定的消息	并发测试	各环节工作正常	pass

3.12.3 测试方法

- 进入顶层目录，执行./script/build.sh 进行系统编译，生成的产物在 out 目录下
- 将生成的 out/目录下的产物推送到目标开发板上，增加执行权限
- 按照测试用例的内容描述进行测试

3.12.4 测试通过条件

- 通过基本的 Demo 功能需求测试，即实现 AB、AC 间的通信流程，且 BC 间无法通信
- 通过大规模并发通信场景测试
- 对出现的的一些异常情况有相应的处理机制

3.12.5 测试结论

系统消息通信功能基本正常，且满足大规模并发通信，对部分的异常情况有相应应对措施。

3.13 系统文件上传功能测试

验证 BC 客户端向服务端上传文件功能是否正常

3.13.1 前置条件

- 源码编译无告警，能正常生成产物，能在目标板上正常运行
- 各模块 netlink 初始化功能正常
- 各公共模块功能正常

3.13.2 测试用例

测试用例	内容描述	备注	通过条件	通过情况
file_upload_test_1	输入客户端目录不存在的文件	异常测试	打印文件不存在，提升重新输入	pass
file_upload_test_2	传输 1k 左右文件	正常测试	文件上传完成，hash 值校验成功	pass
file_upload_test_3	传输 10k 左右文件	正常测试	文件上传完成，hash 值校验成功	pass
file_upload_test_4	传输 100k 左右文件	正常测试	文件上传完成，hash 值校验成功	pass
file_upload_test_5	传输 1M 左右文件	正常测试	文件上传完成，hash 值校验成功	pass
file_upload_test_6	传输 10M 左右文件	正常测试	文件上传完成，hash 值校验成功	pass
file_upload_test_7	传输 100M 左右文件	正常测试	文件上传完成	pass

测试用例	内容描述	备注	通过条件	通过情况
file_upload_test_8	BC 同时传输 100M 左右文件	并发测试	，hash 值校验成功 文件上传完成 ，hash 值校验成功	pass

Table：文件上传测试用例

3.13.3 测试方法

- 进入顶层目录，执行./script/build.sh 进行系统编译，生成的产物在 out 目录下
- 将生成的 out/目录下的产物推送到目标开发板上，增加执行权限
- 加载内核模块，将服务端 A 与客户端 BC 分别放入 out/server out/clientb out/clientc 目录运行
- 将相关测试传输文件放入相应的目录

3.13.4 测试通过条件

- 通过不同大小文件的上传功能测试
- 通过并发的文件上传测试

3.13.5 测试结论

系统文件上传功能正常，可支持多个客户端同时上传文件，且对部分异常情况有相应的处理。

3.14 系统文件下载功能测试

验证 BC 客户端从服务端下载文件功能是否正常

3.14.1 前置条件

- 源码编译无告警，能正常生成产物，能在目标板上正常运行
- 各模块 netlink 初始化功能正常
- 各公共模块功能正常

3.14.2 测试用例

测试用例	内容描述	备注	通过条件	通过情况
file_upload_test_1	输入服务端目录不存在的文件	异常测试	服务端反馈异常消息，文件不存在	fail
file_upload_test_2	传输 1k 左右文件	正常测试	文件下载完成，，hash 值校验成功	pass
file_upload_test_3	传输 10k 左右文件	正常测试	文件下载完成，hash 值校验成功	pass
file_upload_test_4	传输 100k 左右文件	正常测试	文件下载完成，hash 值校验成功	pass
file_upload_test_5	传输 1M 左右文件	正常测试	文件下载完成，hash 值校验成功	pass
file_upload_test_6	传输 10M 左右文件	正常测试	文件下载完成，hash 值校验成功	pass
file_upload_test_7	传输 100M 左右文件	正常测试	文件下载完成，hash 值校验成功	pass
file_upload_test_8	BC 同时传输 100M 左右文件	并发测试	文件下载完成，hash 值校验成功	pass

Table：文件下载测试用例

3.14.3 测试方法

- 进入顶层目录，执行./script/build.sh 进行系统编译，生成的产物在 out 目录下
- 将生成的 out/目录下的产物推送到目标开发板上，增加执行权限
- 加载内核模块，将服务端 A 与客户端 BC 分别放入 out/server out/clientb out/clientc 目录运行
- 将相关测试传输文件放入相应的目录

3.14.4 测试通过条件

- 通过不同大小文件的下载功能测试
- 通过并发的文件下载测试

3.14.5 测试结论

系统文件下载功能正常，可支持多个客户端同时下载文件。
对服务端不存在的文件下载，缺乏异常处理机制。

4 性能测试

4.1 cpu 测试

测试软件 demo 运行各个功能时的 CPU 占用情况。

4.1.1 前置条件

- 源码编译无告警，能正常生成产物，能在目标板上正常运行
- 各模块 netlink 初始化功能正常
- 各公共模块功能正常

4.1.2 测试方法

- 进入顶层目录，执行./script/build.sh 进行系统编译，生成的产物在 out 目录下
- 将生成的目标产物分别运行，测试各个模块在无操作、发送消息、传输文件时的各自 CPU 占用情况

4.1.3 测试通过条件

- 无数据传输是不能占用过高 CPU
- 进行数据传输时，CPU 占比在同平台正常范围内

4.1.4 测试结论

表 4-1: CPU 测试情况

进程名称	场景	CPU 占用率
server_a	无数据传输	0
server_a	发送消息	1%
server_a	传输文件	7%
client_b	无数据传输	0%

进程名称	场景	CPU 占用率
client_b	发送消息	1%
client_b	传输文件	4%
client_b	无数据传输	0%
client_b	发送消息	1%
client_b	传输文件	4%

软件 CPU 占比测试符合要求

4.2 占用内存测试

测试软件 demo 运行各个功能时的内存占用情况。

4.2.1 前置条件

- 源码编译无告警，能正常生成产物，能在目标板上正常运行
- 各模块 netlink 初始化功能正常
- 各公共模块功能正常

4.2.2 测试方法

- 进入顶层目录，执行./script/build.sh 进行系统编译，生成的产物在 out 目录下
- 将生成的目标产物分别运行，测试各个模块在无操作、发送消息、传输文件时的各自内存占用情况

4.2.3 测试通过条件

- 无数据传输是不能占用过高内存
- 进行数据传输时，内存占比在同平台正常范围内

4.2.4 测试结论

表 4-2: 内存占用测试情况

进程名称	场景	实时内存占用
server_a	无数据传输	1460kb
server_a	发送消息	1460kb
server_a	传输文件	2512kb
client_b	无数据传输	2360kb
client_b	发送消息	2430kb
client_b	传输文件	3412kb
client_b	无数据传输	2360kb
client_b	发送消息	2430kb
client_b	传输文件	3412kb

软件内存占比测试符合要求

4.3 硬盘读写测试

测试软件 demo 运行各个功能时的硬盘读写速度情况。

4.3.1 前置条件

- 源码编译无告警，能正常生成产物，能在目标板上正常运行
- 各模块 netlink 初始化功能正常
- 各公共模块功能正常

4.3.2 测试方法

- 进入顶层目录，执行./script/build.sh 进行系统编译，生成的产物在 out 目录下
- 将生成的目标产物分别运行，测试各个模块在无操作、发送消息、传输文件时的各自硬盘读写速度情况

4.3.3 测试通过条件

- 进行数据传输时，硬盘读写速度在同平台正常范围内

4.3.4 测试结论

由于平台工具缺乏，暂时无法测试读写速度，待后续移植相关工具进行测试补充。

5 安全性测试

5.1 内存泄露测试

测试软件 demo 各个状态下是否存在内存泄露

5.1.1 前置条件

- 源码编译无告警，能正常生成产物，能在目标板上正常运行
- 各模块 netlink 初始化功能正常
- 各公共模块功能正常

5.1.2 测试方法

- 进入顶层目录，执行./script/build.sh 进行系统编译，生成的产物在 out 目录下
- 将生成的目标产物分别运行，测试在软件运行前后系统内存情况
- 分析查看生成的 mtrace 日志

5.1.3 测试通过条件

- demo 软件未发生内存泄露

5.1.4 测试结论

软件 demo 没有发生内存泄露

5.2 文件描述符泄露测试

测试软件 demo 各个状态下是否存在文件描述符泄露

5.2.1 前置条件

- 源码编译无告警，能正常生成产物，能在目标板上正常运行
- 各模块 netlink 初始化功能正常
- 各公共模块功能正常

5.2.2 测试方法

- 进入顶层目录，执行./script/build.sh 进行系统编译，生成的产物在 out 目录下
- 将生成的目标产物分别运行，测试在多次文件上传和下载的过程中系统的最大文件描述符的值

5.2.3 测试通过条件

- 在多次文件传输过程之后系统最大文件描述符没有增加

5.2.4 测试结论

软件 demo 没有发生文件描述符泄露

6 测试报告补充说明

6.1 测试用例

本测试报告中列出的测试用例为一类测试用例的分类，实际测试过程中测试用例为单个或多个符合这类描述的用例。

6.2 测试方法

本测试报告采用的是手动测试，因为时间关系存在着完成度不高的问题，自动化测试与用例设计有待后续迭代补充。

7 遗留缺陷

- 封装模块不能对不存在的数据类型、发送者和接收者进行判断并作出相应的处理
- 文件下载过程中，对服务端不存在的文件下载情况，缺乏异常处理机制

著作权声明

商标声明

免责声明