# Android开发中几个好用的工具介绍(1)

# strace工具

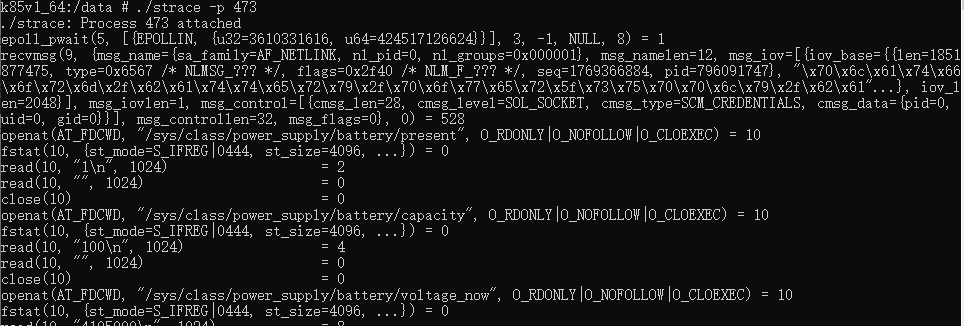
源代码在SDK目录下的external/strace，默认不编译到系统中，直接使用mmma external/strace命令编译即可。

strace介绍网上很多，主要就是可以追踪进程的运行过程和当前状态，以监控电池状态的healthd应用为例：

先查看进程号为473：



strace –p 473看效果：



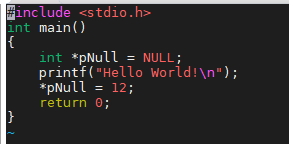
可以看到进程定时的在读取/sys/class/power\_supply/battery/下的相关节点信息。

## 编译android下的c程序

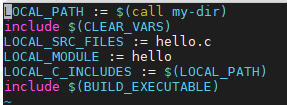
在网上下载arm64的编译器：

gcc-linaro-6.3.1-2017.05-x86\_64\_aarch64-linux-gnu

一个简单的hello.c程序：



一个Android.mk编译规则文件：



编译程序：

./gcc-linaro-6.3.1-2017.05-x86\_64\_aarch64-linux-gnu/bin/aarch64-linux-gnu-gcc -g -o hello -static ./hello.c



这里很重要一点，加上-static编译项，这样不再使用系统的动态库，避免编译器差异引起的问题。

运行看看：



这里故意做了个段错误，是为了后面介绍一个core-dump功能提前做的准备。

## Core-dump功能

Core-dump功能可以在进程崩溃的时候产生core文件，通过解析core文件可以直接找到奔溃的代码位置。这个功能默认是关闭的。

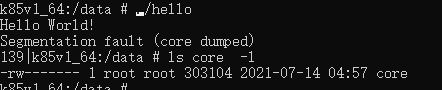
首先打开core-dump功能：

ulimit -c unlimited

查看状态为core，表示开启成功



运行下上面的hello程序：



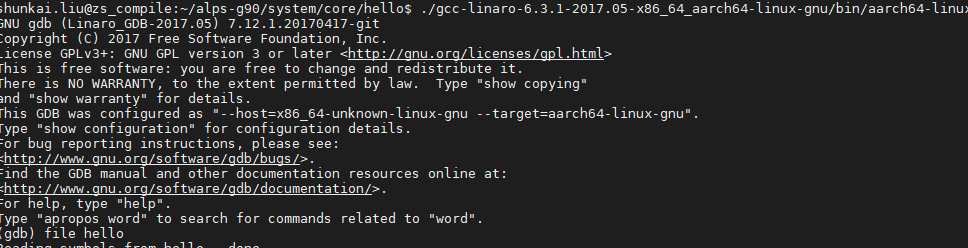
当前目录下产生了一个core文件。

下面解析此文件：

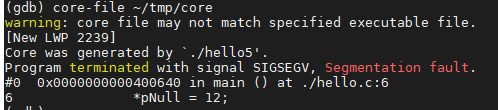
运行gdb:

./gcc-linaro-6.3.1-2017.05-x86\_64\_aarch64-linux-gnu/bin/aarch64-linux-gnu-gdb

并加载hello固件：



加载生成的core文件：



直接显示了异常的语句为hello.c的第6行位置

## 系统crash信息查看

系统crash是个很棘手的问题，但最糟糕的莫过于crash的时候没有接uart口打印内核信息，下次复现就指不定什么时候，费时费力。

Android系统目前提供了一个pstore机制，在不断电的时候，重启系统，可以在/sys/fs/pstore目录下查看到系统最后一段时间内的系统内核信息，当然也包括crash的打印信息。

做个实验：

命令促发 系统奔溃：

echo 'c'>/proc/sysrq-trigger

重启系统，查看



生成了console-ramoops等文件，主要查看这个文件：

可以很清楚的看到crash的函数调用栈,实际测试保存的就是上次开机的最后内核信息，基本上内核信息打印到2秒左右就可以保存到了。

