export PATH=/tmp/openwrt-sdk-19.07.4-ramips-mt7620\_gcc-7.5.0\_musl.Linux-x86\_64/staging\_dir/toolchain-mipsel\_24kc\_gcc-7.5.0\_musl/bin:$PATH

export STAGING\_DIR=/tmp/openwrt-sdk-19.07.4-ramips-mt7620\_gcc-7.5.0\_musl.Linux-x86\_64/staging\_dir/

./configure --build=x86\_64-pc-linux-gnu --host=mipsel-unknown-linux-gnu CC='/tmp/openwrt-sdk-19.07.4-ramips-mt7620\_gcc-7.5.0\_musl.Linux-x86\_64/staging\_dir/toolchain-mipsel\_24kc\_gcc-7.5.0\_musl/bin/mipsel-openwrt-linux-gcc' CFLAGS=-static

##搞完还可以strip一下，缩小体积——800多K到200多K##

export CROSS="mipsel-openwrt-linux-musl-" ; NM="mipsel-openwrt-linux-musl-nm" STRIP="/tmp/openwrt-sdk/staging\_dir/host/bin/sstrip" STRIP\_KMOD="/tmp/openwrt-sdk/scripts/strip-kmod.sh" PATCHELF="/tmp/openwrt-sdk/staging\_dir/host/bin/patchelf" /tmp/openwrt-sdk/scripts/rstrip.sh ./radvd

Cross-compile-ndppd\_static

1)使用padavan工具链只需要在makefile链接处加上-static即可，但是编译出来文件有5M多，就算加上CXXFLAGS += -ffunction-sections -fdata-sections

LDFLAGS += -Wl,-gc-sections也有4.9M，还是太大

1. 使用openwrt-sdk加openwrt-ndppd的Makefile，动态编译很简单就可以通过，但是在padavan上用不了，而静态编译在Makefile的LDFLAGS后面加上-static会报缺libgcc\_s库，因为库里面只有libgcc\_s.so动态库，而没有libgcc\_s.a静态库，后面折腾两天发现可以用/staging\_dir/toolchain-mipsel\_24kc\_gcc-7.5.0\_musl/lib/gcc/mipsel-openwrt-linux-musl/7.5.0/libgcc\_initial.a代替，即在此目录下建立链接ln -sf libgcc\_initial.a libgcc\_s.a即可通过编译并在padavan上运行,而且仅有200多K，可能是链接的库更合理，也可能是openwrt-sdk编译后会有交叉strip工具处理的原因，所以大小更合适，完美！！

～～完美个p，结果运行是可以运行，就是一到iface read（）len=86的ns包就bus error，32和24长度的na包倒是可以读，就是不知道有没有handle转发出去，总之用openwrt-sdk编译试图缩小体积暂时算是失败了

PS:其实折腾时还发现在Makefile的LDFLAGS后面加上-static，然后在LIBS后改为-Wl，-Bstatic,-lc,-luClibc++ -Wl,-Bdynamic,-lgcc\_s也可以通过编译实现半静态编译，但是在padavan上不能运行，怀疑要么是padavan上的libgcc\_s.so库不匹配，要么是-Bdynamic放到后面导致还有什么别的库动态链接了，而padavan上没有这个库，可以佐证的是在-Wl，-Bstatic,-lc,-luClibc++ -Wl,-Bdynamic,-lgcc\_s后再加-Wl，-Bstatic后或是-Wl,-Bdynamic,-lgcc\_s放在-Wl，-Bstatic,-lc,-luClibc++前面，编译就无法通过，还是提示找不到gcc\_s

PS+1：又折腾两天发现（1）用上述-Wl，-Bstatic，-lstdc++,-Bdynamic方式在padavan的toolchain下编译的也运行不了，结论应该是不管怎么调整，都存在库的相互依赖顺序导致不能实现该方式的半静态编译（而且需要注意的是在动态编译下不同的编译器musl和uclibc的会导致链接的动态库不一样，所以上述参数-l库名也有不同，具体可以动态编译后传上padavan然后用ldd查看），但是对于padavan的toolchain用的libstdc++可以对g++用-static-libstdc++参数实现libstdc++库静态编译，而其他库动态编译，只是这样出来的也是5M多，只少大概一二百K吧

1. 也正因为上述原因，加上在openwrt-sdk下用libgcc\_initial.a代替编译出来的运行时始终看不到handle ns na的输出，而且遇到86长度的ns包就bus error，所以有理由怀疑libgcc\_initial.a库不能替代libgcc\_s，要么是还有函数没有链接进来，而openwrt动态编译后可以以很小的libgcc\_s动态库运行，但padavan静态链接了stdc++库就有那么大，所以可能是openwrt的内核还有些别的库在帮忙
2. 还有在padavan的toolchain下用-static-libgcc参数可以把libgcc\_s.so的依赖去掉，但同时在x86主机上用-static-libgcc直接编译时和在openwrt-sdk下一样无法去掉libgcc\_s.so的依赖，说明主机和openwrt-sdk下的libgcc.a库相比libgcc\_s.so还是缺了些东西，这也是为什么在openwrt-sdk中ln -sf libgcc.a libgcc\_s.a编译会包缺函数或缺定义，但padavan的toolchain里面libgcc.a有10M多（相比openwrt-sdk里的只有1M多），就完全涵盖了libgcc\_s.so的函数，但是拿到openwrt-sdk里也用不了，大约是版本不同（uclibc和musl的区别）或是因为之前编译它们的编译器不同，所以函数的定义不一样了（因为拷到openwrt-sdk想代替libgcc\_s.a，结果跟用openwrt-sdk自己的libgcc.a代替效果差不多，也是包缺函数或缺定义）
3. 一个基本点：好像就不存在libgcc\_s.a，gcc那边准备给的就是有libgcc.a的静态库和libgcc\_s.so的动态库（但是同时又存在libgcc.so??），所以libgcc\_s就是用来动态链接的

Damn it！！搞了好多天，就为了把ndppd体积缩小，结果padavan的toolchain编译的原本5.6M，压缩完还有1.9M，用上文提到的命令strip一下就变成1.1M，压缩完才400K，早知道就不用折腾了

没仔细去找交叉版upx，upx一下说不定能更小