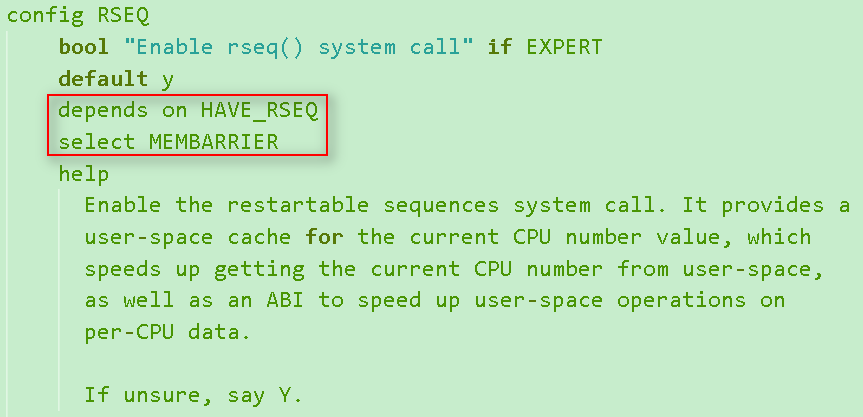
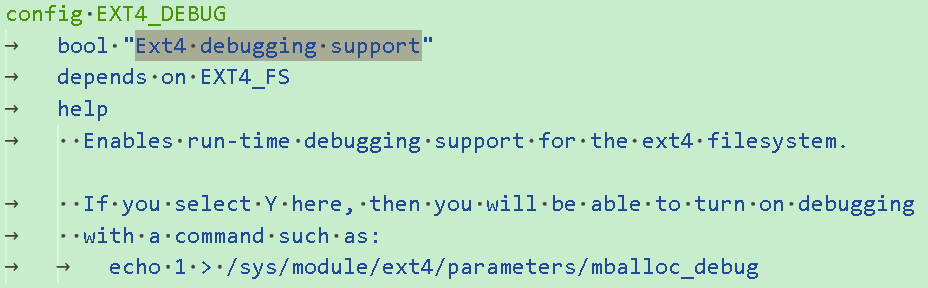
# 配置项选择思路

* 所管理的代码量小一点的
* 尽量找不depend或者select过多其他配置项的
* 定义了新的全局变量或者在全局结构体变量加了新成员的

例如：Enable rseq() system call配置项既依赖于HAVE\_RSEQ配置，并且会自动开启MEMBARRIER配置，不便于统计。



# file systems->ext4 debugging support



# 配置项管理的源码

**#ifdef** CONFIG\_EXT4\_DEBUG

        ext4\_warning(inode->i\_sb, "Inode (%ld) finished: extent logical block %llu,"

                 " len %u; IO logical block %llu, len %u",

                 inode->i\_ino, (**unsigned** **long** **long**)ee\_block, ee\_len,

                 (**unsigned** **long** **long**)map->m\_lblk, map->m\_len);

**#endif**

**#ifdef** CONFIG\_EXT4\_DEBUG

**ushort** ext4\_mballoc\_debug \_\_read\_mostly;

module\_param\_named(mballoc\_debug, ext4\_mballoc\_debug, **ushort**, **0**644);

MODULE\_PARM\_DESC(mballoc\_debug, "Debugging level for ext4's mballoc");

**#endif**

**#ifdef** CONFIG\_EXT4\_DEBUG

**static** **void** ext4\_mb\_show\_ac(**struct** ext4\_allocation\_context \*ac)

{

**struct** super\_block \*sb = ac->ac\_sb;

    ext4\_group\_t ngroups, i;

**if** (!ext4\_mballoc\_debug ||

        (EXT4\_SB(sb)->s\_mount\_flags & EXT4\_MF\_FS\_ABORTED))

**return**;

    ext4\_msg(ac->ac\_sb, KERN\_ERR, "Can't allocate:"

            " Allocation context details:");

    ext4\_msg(ac->ac\_sb, KERN\_ERR, "status %d flags %d",

            ac->ac\_status, ac->ac\_flags);

    ext4\_msg(ac->ac\_sb, KERN\_ERR, "orig %lu/%lu/%lu@%lu, "

            "goal %lu/%lu/%lu@%lu, "

            "best %lu/%lu/%lu@%lu cr %d",

            (**unsigned** **long**)ac->ac\_o\_ex.fe\_group,

            (**unsigned** **long**)ac->ac\_o\_ex.fe\_start,

            (**unsigned** **long**)ac->ac\_o\_ex.fe\_len,

            (**unsigned** **long**)ac->ac\_o\_ex.fe\_logical,

            (**unsigned** **long**)ac->ac\_g\_ex.fe\_group,

            (**unsigned** **long**)ac->ac\_g\_ex.fe\_start,

            (**unsigned** **long**)ac->ac\_g\_ex.fe\_len,

            (**unsigned** **long**)ac->ac\_g\_ex.fe\_logical,

            (**unsigned** **long**)ac->ac\_b\_ex.fe\_group,

            (**unsigned** **long**)ac->ac\_b\_ex.fe\_start,

            (**unsigned** **long**)ac->ac\_b\_ex.fe\_len,

            (**unsigned** **long**)ac->ac\_b\_ex.fe\_logical,

            (**int**)ac->ac\_criteria);

    ext4\_msg(ac->ac\_sb, KERN\_ERR, "%d found", ac->ac\_found);

    ext4\_msg(ac->ac\_sb, KERN\_ERR, "groups: ");

    ngroups = ext4\_get\_groups\_count(sb);

**for** (i = 0; i < ngroups; i++) {

**struct** ext4\_group\_info \*grp = ext4\_get\_group\_info(sb, i);

**struct** ext4\_prealloc\_space \*pa;

        ext4\_grpblk\_t start;

**struct** list\_head \*cur;

        ext4\_lock\_group(sb, i);

        list\_for\_each(cur, &grp->bb\_prealloc\_list) {

            pa = list\_entry(cur, **struct** ext4\_prealloc\_space,

                    pa\_group\_list);

            spin\_lock(&pa->pa\_lock);

            ext4\_get\_group\_no\_and\_offset(sb, pa->pa\_pstart,

                             NULL, &start);

            spin\_unlock(&pa->pa\_lock);

            printk(KERN\_ERR "PA:%u:%d:%u \n", i,

                   start, pa->pa\_len);

        }

        ext4\_unlock\_group(sb, i);

**if** (grp->bb\_free == 0)

**continue**;

        printk(KERN\_ERR "%u: %d/%d \n",

               i, grp->bb\_free, grp->bb\_fragments);

    }

    printk(KERN\_ERR "\n");

}

**#else**

**#ifdef** CONFIG\_EXT4\_DEBUG

**extern** **ushort** ext4\_mballoc\_debug;

**#define** mb\_debug(n, fmt, ...)                                           \

**do** {                                    \

**if** ((n) <= ext4\_mballoc\_debug) {                \

        printk(KERN\_DEBUG "(%s, %d): %s: " fmt,         \

               \_\_FILE\_\_, \_\_LINE\_\_, \_\_func\_\_, ##\_\_VA\_ARGS\_\_);    \

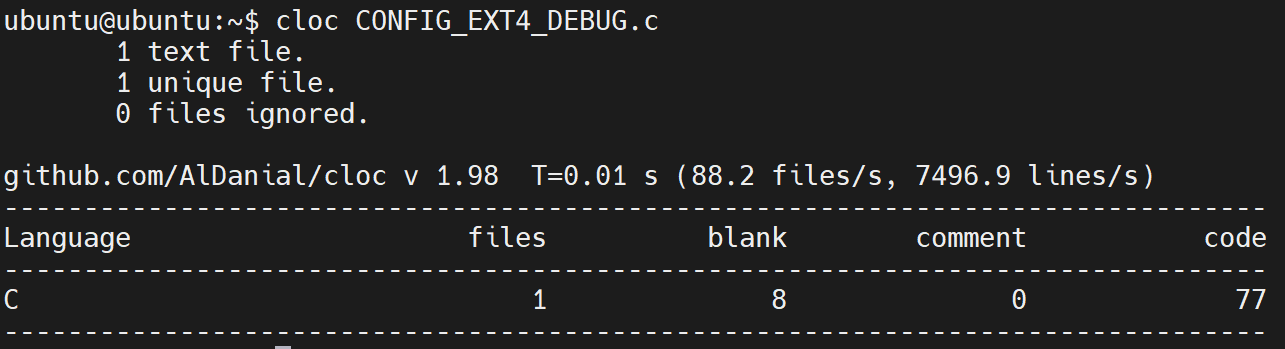
    }                               \

} **while** (0)

**#else**

# 源码分析

新增代码行数：77



读写语句行数占新增代码行数比例：2/77

有两句对ext4\_mballoc\_debug全局变量的读操作

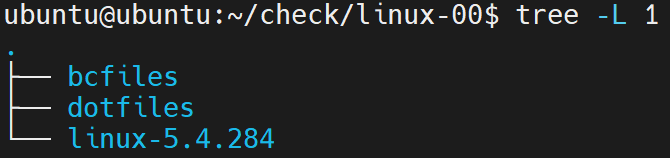
**if** (!ext4\_mballoc\_debug ||

        (EXT4\_SB(sb)->s\_mount\_flags & EXT4\_MF\_FS\_ABORTED))

和

**if** ((n) <= ext4\_mballoc\_debug) {                \

# CFG分析



* bcfiles：存储所有bc文件
* dotfiles：存储所有dot文件

# 先获得默认配置的linux的bc和dot文件

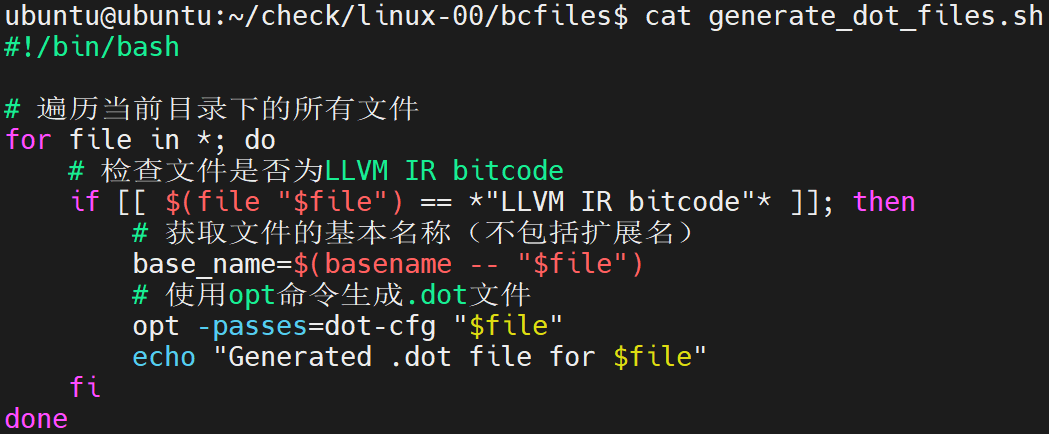
export WLLVM\_BC\_STORE=/pathto/bcfiles

export LLVM\_COMPILER=clang

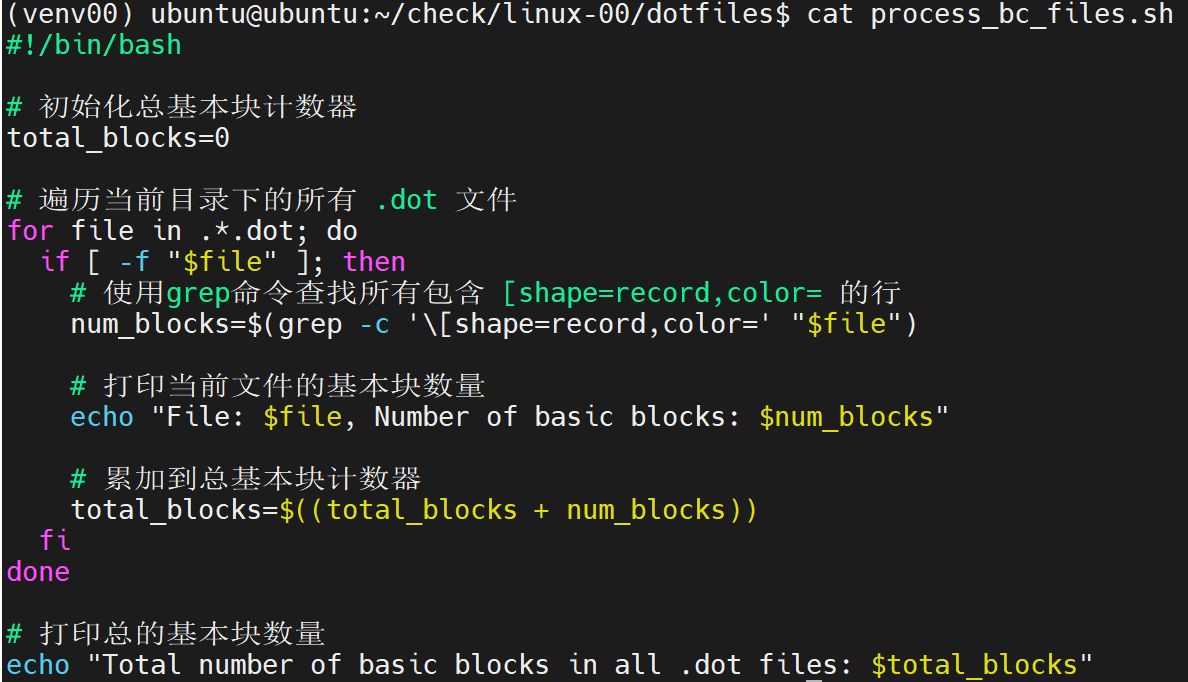
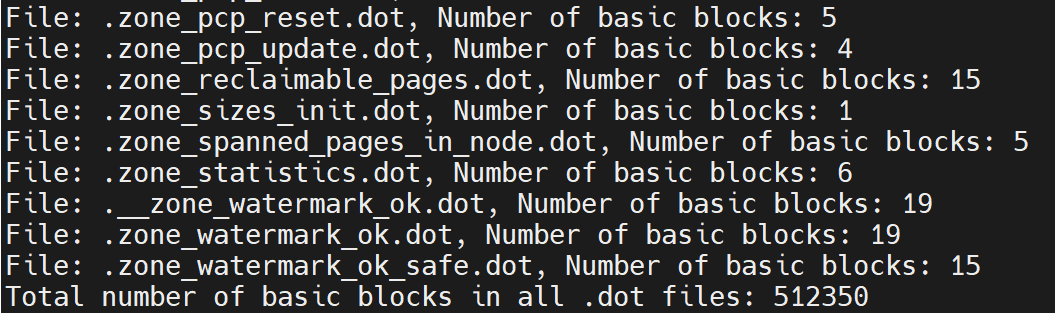
make CC=wllvm defconfig

make CC=wllvm LLVM=1 -j8

在bcfiles下运行下面脚本，用来生成dot文件



生成dot文件后，移到dotfiles下，运行下面脚本，做统计

# 打开ext4 debugging support配置项，重新编译

make CC=wllvm menuconfig

make CC=wllvm LLVM=1 -j8

# 再生成dot文件，并统计