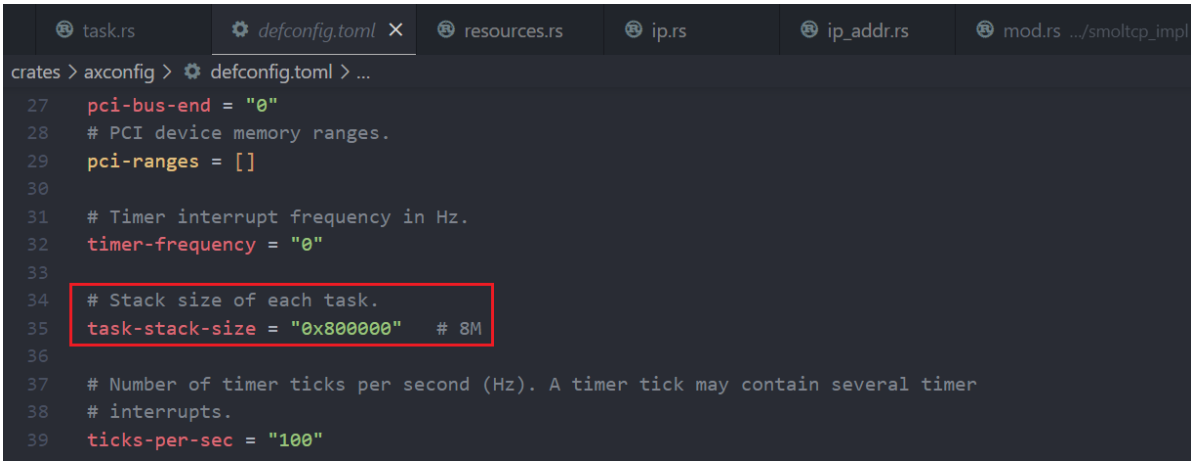
任务介绍

13. prlimit 需要支持设置栈大小 （2周，8.9）
    - 目前的 `syscall_prlimit64` 仅支持修改最大文件数，用户栈大小固定是 256K，但 ffmpeg 的调用是 `` `prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0` ``直接要了 8M 的栈大小。这可能需要动一下 TCB 的结构，把常量改成变量。可以参考 `curr_process.fd_manager.set_limit(new_limit)` 设置文件数，写一个类似 `curr_process.set_stack_limit` 的东西。

源代码

```
pub fn syscall_prlimit64(args: [usize; 6]) -> SyscallResult {
    let pid = args[0];
    let resource = args[1] as i32;
    let new_limit = args[2] as *const RLimit;
    let old_limit = args[3] as *mut RLimit;
    // 当pid不为0，其实没有权利去修改其他的进程的资源限制
    let curr_process = current_process();
    error!("TASK_STACK_SIZE is {}", TASK_STACK_SIZE);
    if pid == 0 || pid == curr_process.pid() as usize {
        match resource {
            RLIMIT_STACK => {
                if old_limit as usize != 0 {
                    unsafe {
                        *old_limit = RLimit {
                            rlim_cur: TASK_STACK_SIZE as u64,
                            rlim_max: TASK_STACK_SIZE as u64,
                        };
                    }
                }
            }
```

TASK_STACK_SIZE是个常量，代表栈大小。可以在/crate/axconfig/defconfig.toml中修改



由于只在ffmpeg中用到这个特定大小栈，不必将其改为变量，可以在/crates/linux_syscall_api/src/ctypes.rs再添加两个常量来实现定义该栈的大小

修改过后的代码

/crate/linux_syscall_api/src/syscall_task/imp/task.rs

```
pub fn syscall_prlimit64(args: [usize; 6]) -> SyscallResult {
    let pid = args[0];
    let resource = args[1] as i32;
    let new_limit = args[2] as *const RLimit;
    let old_limit = args[3] as *mut RLimit;
    // 当pid不为0，其实没有权利去修改其他的进程的资源限制
    let curr_process = current_process();

if pid == 0 || pid == curr_process.pid() as usize {
    match resource {
        RLIMIT_STACK => {
            if old_limit as usize != 0 {
                if old_limit as usize == APPLY_STACK_SIZE {
                    unsafe {
                        *old_limit = RLimit {
                            rlim_cur: FFMPEG_STACK_SIZE as u64,
                            rlim_max: FFMPEG_STACK_SIZE as u64,
                        };
                    }
                } else {
                    unsafe {
                        *old_limit = RLimit {
                            rlim_cur: TASK_STACK_SIZE as u64,
                            rlim_max: TASK_STACK_SIZE as u64,
                        };
                    }
                }
            }
        }
```

/crates/linux_syscall_api/src/ctypes.rs

```
/// 申请使用的栈大小
pub const APPLY_STACK_SIZE: usize = 0x3ffff830;
/// ffmpeg 使用的栈大小
pub const FFMPEG_STACK_SIZE: usize = 0x800000;
```

实现效果

```
        16:     file=libdatrie.so.1 [0];  needed by /lib/libthai.so.0 [0]
        16:     file=libdatrie.so.1 [0];  generating link map
        16:       dynamic: 0x000000000cfb7d40  base: 0x000000000cfaf000   size: 0x0000000000009010
        16:         entry: 0x000000000cfb1220  phdr: 0x000000000cfaf040  phnum:                 11
        16:
[ 43.278561 0:17 linux_syscall_api::syscall_task::imp::task:374] old_limit as usize is 1073739824
[ 43.284307 0:17 linux_syscall_api::syscall_task::imp::task:375] FFMPEG_STACK_SIZE is 8388608
[ 43.290606 0:17 linux_syscall_api::syscall_task::imp::task:383] rlim_cur is 8388608
[ 43.295156 0:17 linux_syscall_api::syscall_task::imp::task:385] set over.
        16:
        16:     calling init: /lib64/ld-linux-x86-64.so.2
        16:
```