Implementace PROC FS

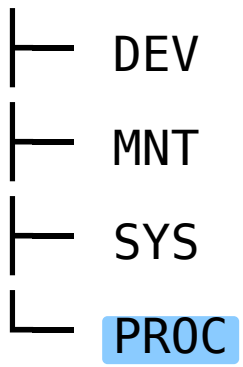
Vladan Trhlík

Cíl

- vytvoření procfs mount point v dosavadním FS KIV-RTOS
- musí obsahovat:
  - aktuálních počet tasků
  - informace plánovače
  - celkový počet otevřených souborů
  - informace ke každému tasku (PID, počet otevřených souborů…)

Struktura

Přidání mount pointu



Složky procesů

- `PROC`
  - `1/`
  - …
  - `[pid]`/
    - `state` - stav tasku (runnable, blocked atd)
    - `fd_n` - počet otevřených souborů
    - `fd` - otevřené soubory
    - `pid` - PID tasku
    - `page` - počet alokovaných stránek
    - `status` - shrnutí stavu (human readable)
  - `self/`
  - …

Informace o systému

- `PROC`
  - ...
  - `sched` - počet runnable, blocked tasků
  - `tasks` - aktuální počet tasků
  - `ticks` - počet tiků od startu (obodoba `/proc/uptime`)

Implementace

Mount point

- `fs/filesystem.h` – `CFileSystem`:

```
class CFilesystem {
    ...
private:
    TFS_Tree_Node mRoot_Dev;
    TFS_Tree_Node mRoot_Sys;
    TFS_Tree_Node mRoot_Mnt;
    TFS_Tree_Node mRoot_Proc;
}
```

FS Driver

- `fs/drivers/proc_fs.h` – třída driveru

```
class CProc_FS_Driver : public IFilesystem_Driver {
    IFile* Open_File(const char* path, NFile_Open_Mode mode)
    { ... }
}
```

Soubor Tasku

```
class CProcFS_PID_File final : public IFile {
    CProcFS_PID_File(int pid, NProcFS_PID_Type type) { ... }
    uint32_t Read(char* buffer, uint32_t num) {
        auto task = sProcessMgr.Get_Process_By_PID(_pid);
    }
}

enum NProcFS_PID_Type {
    PID, STATE, FD_N, FD, STATUS, PAGE
};
```

Soubor systémové informace

```
class CProcFS_Status_File final : public IFile
    CProcFS_Status_File(NProcFS_Status_Type type) { ... }
    uint32_t Read(char* buffer, uint32_t num) {
        sProcessMgr.Get_Scheduler_Info(..., &info);
    }
}

enum NProcFS_Status_Type {
    SCHED, TASKS, TICKS
};
```

děkuji za pozornost