FX-RTOS

EREMEX

Использование уровней привилегий и защиты памяти в ОСРВ для микроконтроллеров

Дмитрий Алексеев

24 июня 2022

OS:DAY

www.eremex.ru

1. Актуальность темы
2. Поддержка ОС архитектурой ARMv7-M
3. Механизмы безопасности ARMv7-M
4. Концепция процессов в FX-RTOS Nanokernel Secure
5. Источники ошибок и атак на kernel-mode
6. Характеристики FX-RTOS Nanokernel Secure
7. Иллюстрация системного вызова
8. Пример программы инициализации

Актуальность

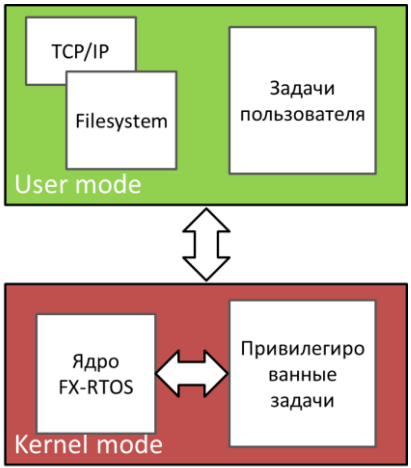
- Распространение IoT устройств;
- Программы содержат уязвимости;
- Отсутствует изоляция процессов;

Общий подход:

- Регионы памяти вместо процессов: изоляция групп задач;
- Защита критически важного кода от остального;
- Программная реакция на вторжения и ошибки;

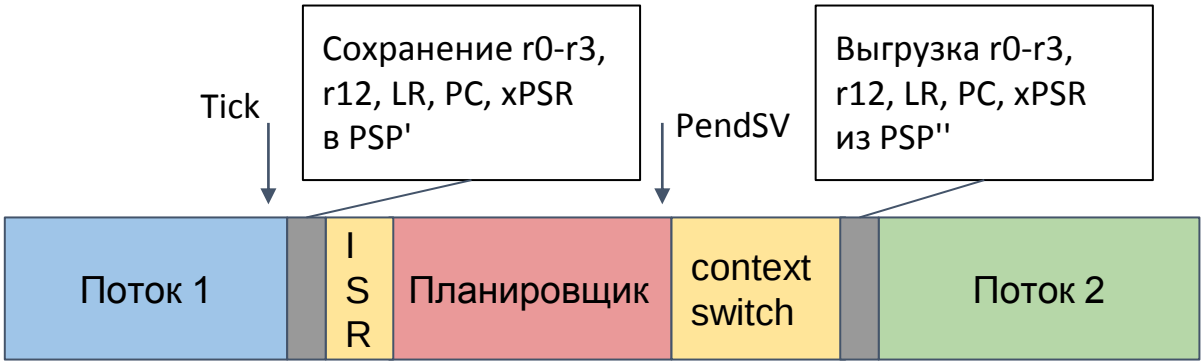
Реализации:

- Двухъядерный МК (AMP)
- RTOS + Интерпретатор (Lua, Micropython, Pawn);
- Поддержка usermode в ОСРВ.

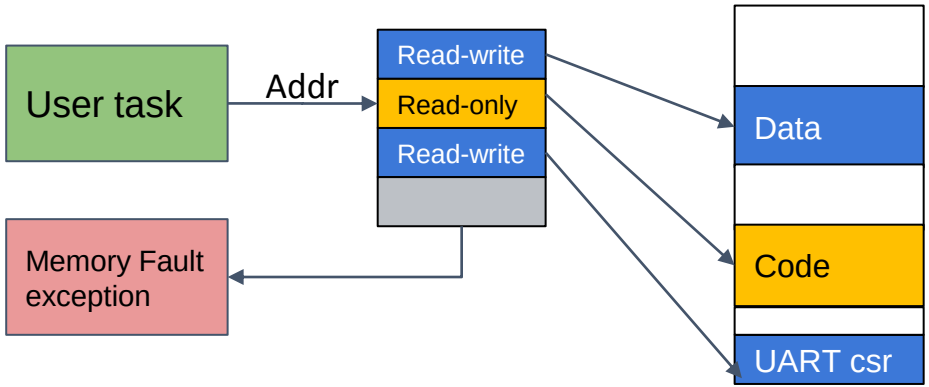


Общая поддержка ОС в ядрах Cortex-M:

- Двойной указатель стека (Main, Process)
- Исключение PendSV
- Аппаратное сохранение/загрузка основных регистров при входе/выходе из прерывания
- Инструкции эксклюзивного доступа к памяти (LDREX, STREX)
- Системный таймер (Systick)

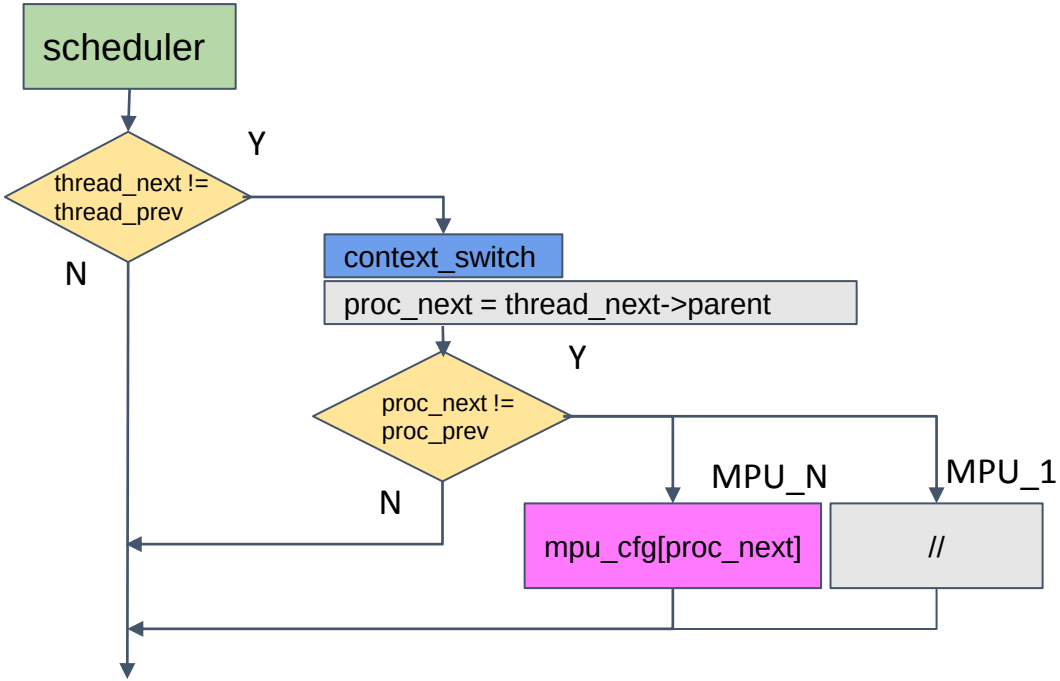


- Два уровня привилегий
- Инструкция и исключение SVC
- Блок защиты памяти MPU:
  - 8 независимо определяемых регионов (16 в ARMv8-M)
    - Чтение
    - Запись
    - Исполнение
    - Кэшируемость (WB, WT, UC)
  - Инициализация перед использованием
  - Может быть перепрограммирован во время работы
  - Ошибка доступа - исключение Memory Management Fault



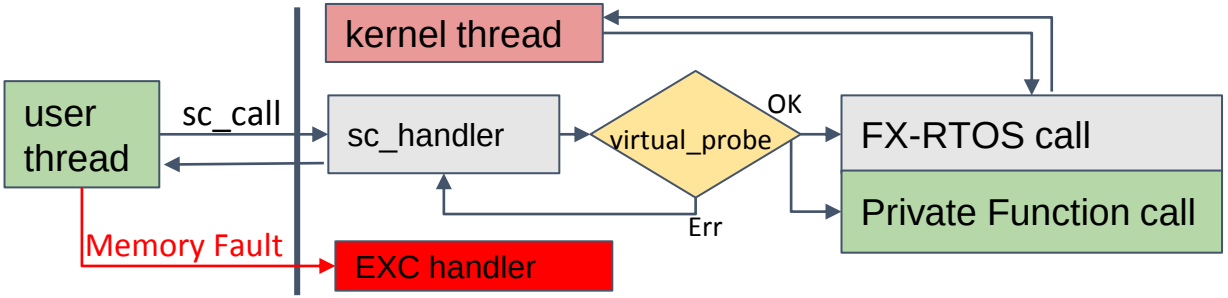
Два альтернативных варианта:

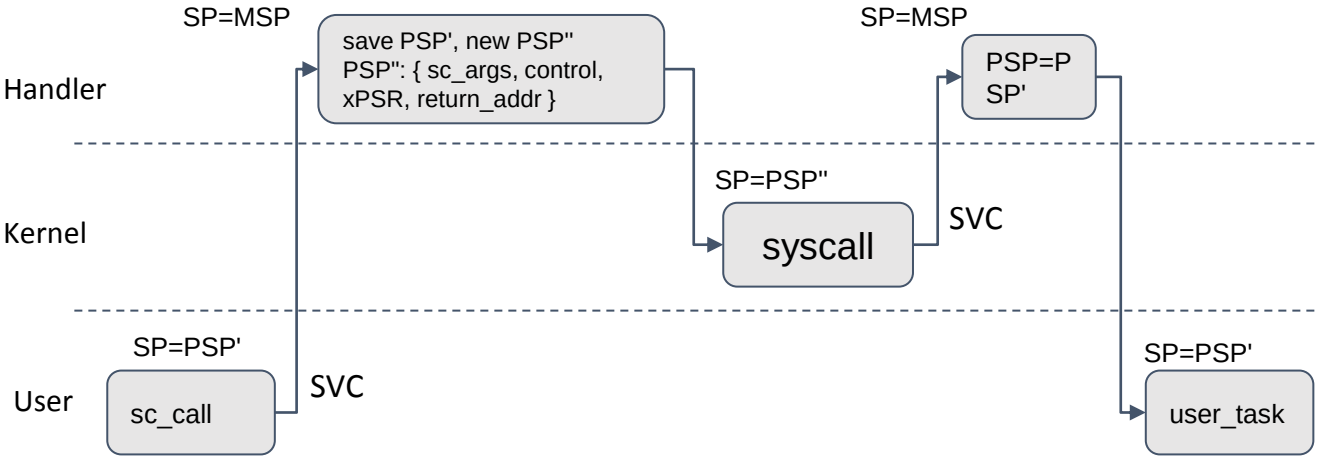
- 2 режима (kernel/user), M процессов : N потоков;
- 2 режима (kernel/user), 1 процесс, N потоков;



- Израсходование памяти в куче
  - Отказ задач, использующих динамическую память
- Переполнение буфера
  - Повреждение данных в памяти
- Ошибки в проектировании системных вызовов
  - Эскалация привилегий
- Прерывания
  - Обработчик в привилегированном режиме
  - Отказ в обслуживании
- Ошибки в драйверах
  - Повреждение данных через DMA

- ОСРВ для микроконтроллеров;
  - Статическая библиотека для организации многозадачности;
- Разделение приложения на привилегированную (доверенную) и не привилегированную часть;
  - 2 бинарных образа kernel/user;
- Совместимость с остальными конфигурациями FX-RTOS;
  - User API - подмножество основного API для приложений;
- Вызов функций ОС через механизм системных вызовов;
  - Контроль доступа к объектам;
  - Контроль указателей;
  - Контроль точек входа ядра;
  - Возможность добавления пользовательских вызовов;
- Минимальные требования;
  - 15 КБ ROM;
  - 10 КБ RAM;
- Переносимость на другие архитектуры МК
  - RISC-V rv32 + PMP


kernel thread
user thread
sc_call
sc_handler
virtual_probe
OK
Err
FX-RTOS call
Private Function call
Memory Fault
EXC handler


SP=MSP
save PSP', new PSP''
PSP'': { sc_args, control,
xPSR, return_addr }
SP=MSP
PSP=P
SP'
Handler
Kernel
SP=PSP''
syscall
SVC
SP=PSP'
User
sc_call
SVC
SP=PSP'
user_task

Пример программы инициализации

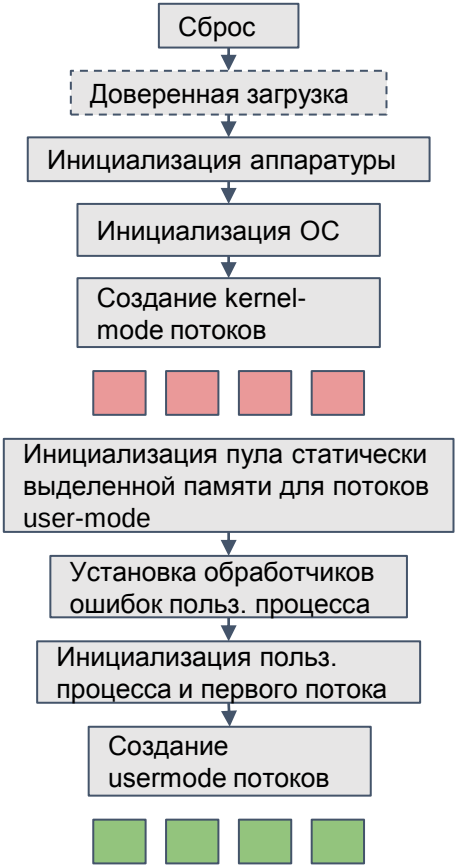
```
void fx_app_init(void)
                    kernel
{
    ex_obj_pool_init(&pool, buf, sizeof(buf));
    ex_process_init(&usr_pcs, &pool, NULL);
    fx_process_set_exception_ex(&usr_pcs.kprocess,
          FX_EXCEPTION_SEGV, exc_handler, NULL);
    ex_process_start(&usr_pcs, usr_entry, NULL,
          30, &_USR_STACK_BASE, &_USR_STACK_SIZE);
}
int main(void)
{
    mpu_init();
    fx_kernel_entry();
}
```

libfxrtos.a
FXRTOS.h

```
void usr_main(void)
                                    user
{
    static fx_thread_t t1;
    static int stk1[2048/sizeof(int)];

    fx_thread_init(&t1, fn, NULL, 20, stk1,
                   sizeof(stk1), false);
    for (;;)
    {
        fx_thread_sleep(100);
    }
}
```

libfxrtos_u.a
FXRTOS_LIB.h



eremex.ru/products/fx-rtos

github.com/Eremex

t.me/fxrtos

+7 (495) 232-1864

info@eremex.ru