# Конструирование ядра операционных систем (X)

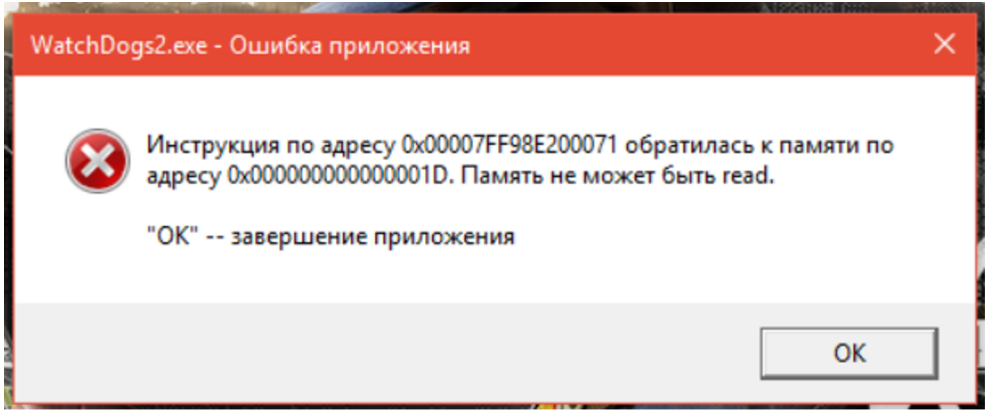
## Обработка исключений в приложениях

# План

- Аппаратные исключения
  - На POSIX-системах
  - Windows SEH / 32-bit
  - Windows SEH / 64-bit
  - Windows VEH
- Программные исключения

# Проблема

```
#include <stdio.h>

int main()
{
    char* str = (void*) 0x1D;
    printf("String at 0x1D is %s\n", str);

    return 0;
}
```



# POSIX сигналы

```
#include <signal.h>
#include <stdio.h>


static void handler(int sig, siginfo_t *si, void *unused)
{
   // handle error
}


int main()
{
   struct sigaction sa;


   sa.sa_flags = SA_SIGINFO;
   sigemptyset(&sa.sa_mask);
   sa.sa_sigaction = handler;
   // segmenattion fault handler
   sigaction(SIGSEGV, &sa, NULL);

   char* str = (void*) 0x1D;
   printf("String at 0x1D is %s\n", str);

   return 0;
}
```

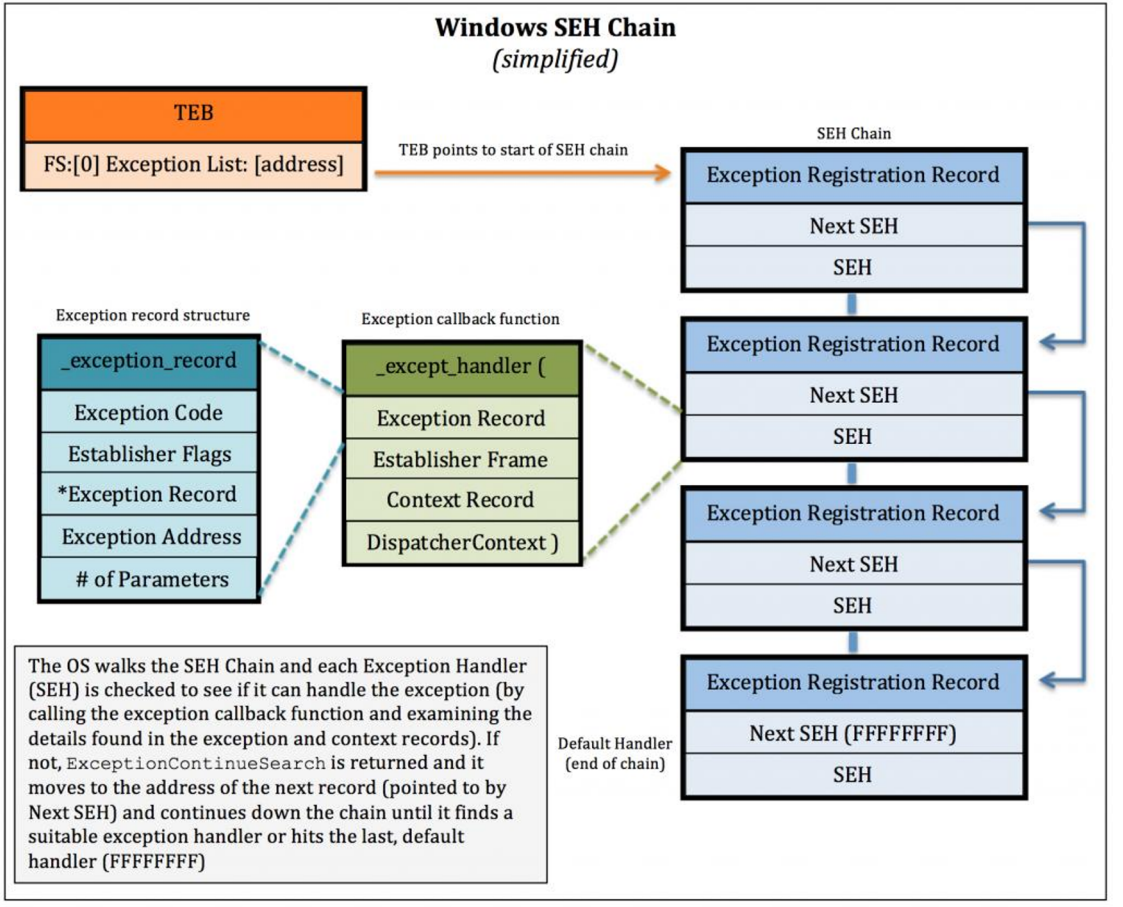
# Windows (SEH)

```
#include <stdio.h>
#include <windows.h> // for EXCEPTION_ACCESS_VIOLATION
#include <excpt.h>

int filter(unsigned int code, struct _EXCEPTION_POINTERS *ep)
{
   if (code == EXCEPTION_ACCESS_VIOLATION)
      return EXCEPTION_EXECUTE_HANDLER;
   else
      return EXCEPTION_CONTINUE_SEARCH;
}

int main()
{
   __try
   {
      char* str = (void*) 0x1D;
      printf("String at 0x1D is %s\n", str);
   }
   __except(filter(GetExceptionCode(), GetExceptionInformation()))
   {
      printf("Exception occured\n");
   }

   return 0;
}
```

- SEH – Structured Exception Handling

# SEH: Устройство (32-bit)

```
__try
{
    char* str = (void*) 0x1D;
    printf("String at 0x1D is %s\n", str);
}
__except(filter(...))
{
    printf("Exception occured\n");
}
```



# SEH: Устройство (64-bit)

- Стек больше не используется.
- Информация о блоках исключений хранится в специальной секции в исполняемом файле (.pdata).

```
typedef struct _RUNTIME_FUNCTION {
    UINT32 BeginAddress;    // Отн. адрес функции, использующей SEH
    UINT32 EndAddress;      // Отн. адрес конца такой функции
    UINT32 UnwindData;      // Адрес структуры, описывающей SEH блоки внутри
} RUNTIME_FUNCTION, *PRUNTIME_FUNCTION;
```

# Vectored Exception Handling

- Механизм, подобный обработчикам сигналов в POSIX.
- Обрабатывает SEH исключения на уровне всего процесса.
- AddVectoredExceptionHandler(CALL_FIRST,VectoredHandlerFunc);

# Программные исключения

- Windows: всегда можно вызвать функцию RaiseException.
- POSIX: отправка сигналов также доступна программно.
- C++ и другие языки с поддержкой исключений могут использовать эти возможности в реализации своей runtime библиотеки.
- C99: setjmp/longjmp.

Спасибо за внимание!

Вопросы?