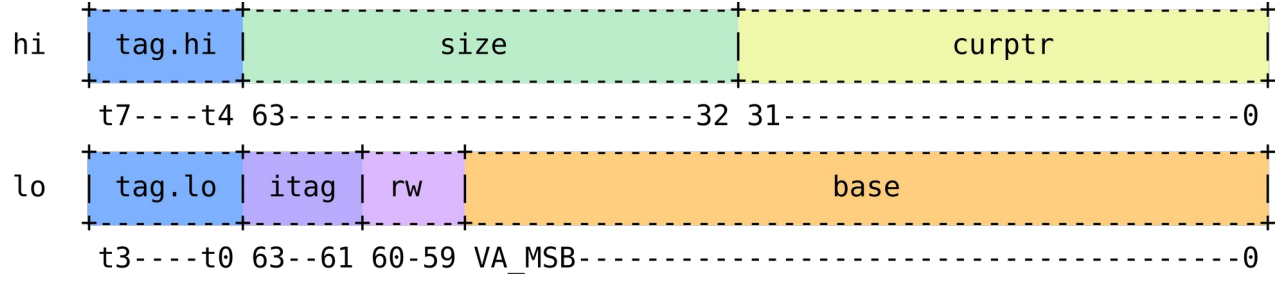
Московский физико-технический институт (национальный исследовательский университет)
Физтех-школа радиотехники и компьютерных технологий
Кафедра информатики и вычислительной техники

Портирование программных средств сборки и тестирования приложений в режим безопасных вычислений на платформе “Эльбрус”

Студент: Титов Е.М.
Научный руководитель: Ярапов Д.В.

Особенности режима безопасных вычислений (РБВ) на платформе “Эльбрус”

- Дополнительный контроль за исполнением программ на языках C и C++.
- Адресация 128-битными дескрипторами, которые содержат:
  - указатель на начало выделенного фрагмента памяти — 64 бита (используется 48 бит);
  - размер этого фрагмента — 32 бита;
  - смещение относительно его начала — 32 бита.
- Выравнивание дескрипторов на 16 байт.
- Подтверждение дескриптора аппаратными тегами.



Особенности режима безопасных вычислений (РБВ) на платформе “Эльбрус”

- Теги располагаются отдельно от основных данных и маркируют числовые данные, дескрипторы и неинициализированные данные.
- Теги дескрипторов сохраняются только при атомарной выровненной записи.
- Дескриптор можно получить только:
  - от ядра операционной системы, например, при помощи *malloc;*
  - взятием адреса данных на стеке или в области глобальных переменных;
  - из другого дескриптора без увеличения размера фрагмента.
- Выполняется аппаратный контроль:
  - обращения за границы объектов;
  - использования неинициализированных переменных;
  - создания и преобразования указателей.

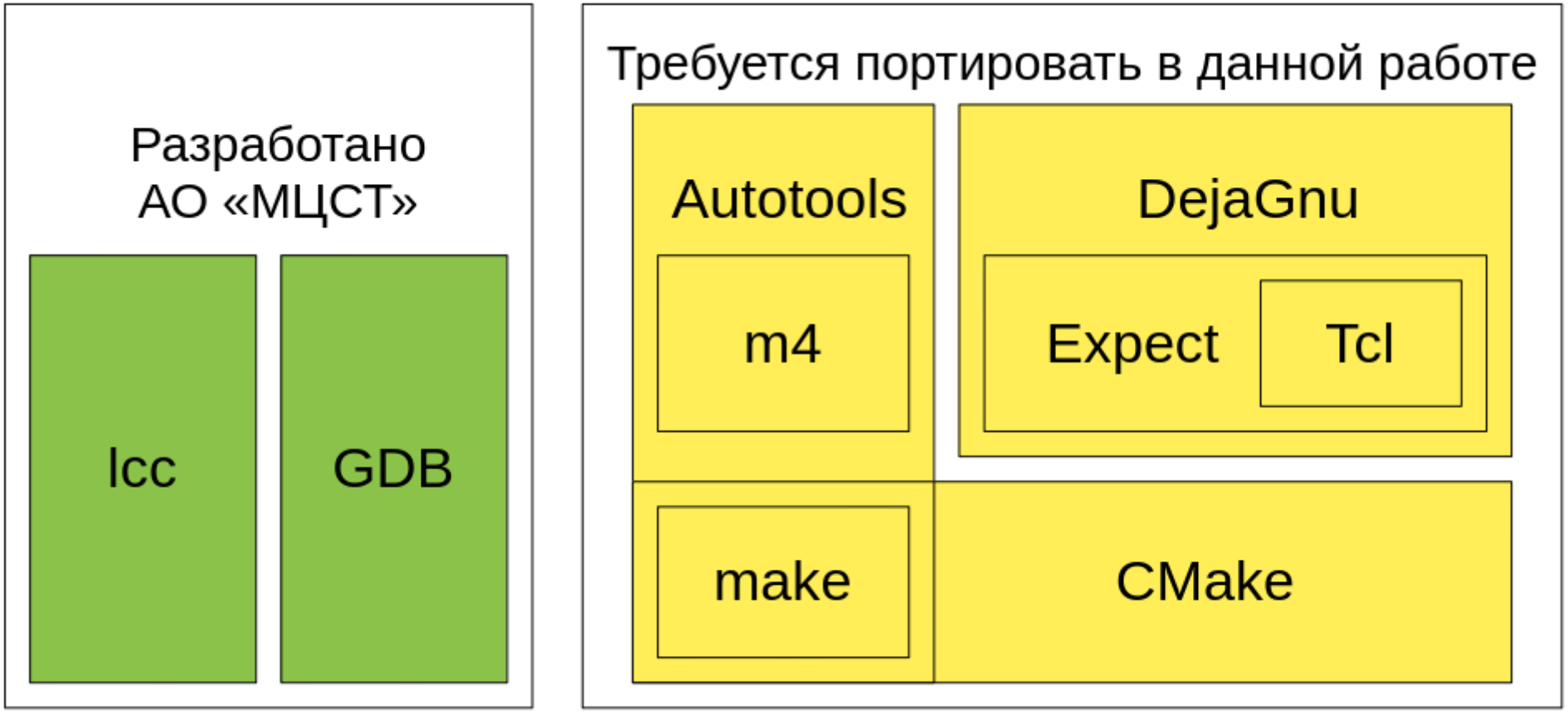
Среда режима безопасных вычислений

- Компиляция в РБВ осуществляется с опцией *-mptr128.*
- Для корректной работы программы в РБВ в неё бывает необходимо вносить изменения — портировать.
- Программы часто имеют зависимости: вызывают другие программы и используют разделяемые библиотеки. Зависимости тоже должны работать в РБВ.
- В АО «МЦСТ» разработана среда, где все компоненты работают в РБВ — контейнер РБВ.
- Для работы с программным пакетом в этой среде необходимо иметь средства сборки и тестирования.

Средства разработки программного обеспечения

- **LCC** — компилятор, разработанный АО «МЦСТ».
- **GDB** — отладчик, работающий с режимом безопасных вычислений.
- GNU **make** — система описания процесса сборки программных проектов.
- GNU **Autotools** — система обеспечения переносимости программ между UNIX-подобными системами. Генерирует сценарии make. Зависимости:
  - GNU **m4** — процессор макросов.
- **CMake** — альтернатива GNU Autotools. Генерирует сценарии make и других систем сборки. Включает в себя средство тестирования **CTest**.
- **DejaGnu** – фреймворк тестирования. Зависимости:
  - **Expect** - средство автоматизации работы с интерактивными приложениями. В свою очередь написан на
    - **Tcl** — языке программирования Tool Command Language.

Средства разработки программного обеспечения



Фаззинг-тестирование программного обеспечения

- На вход программе и в качестве аргументов подаются неожиданные, зачастую некорректные случайные данные.
- Тестируемая программа все равно должна корректно их обработать.
- Фаззинг незаменим в случае отсутствия поставляемых с программой тестов и может покрывать то, что они не покрыли.
- Разрабатывается фаззер **elfuzz** на основе **iFUZZ**, который:
  - Автоматически тестирует все программы в указанной директории.
  - Формирует аргументы, запускает с ними программу, при необходимости завершает её по тайм-ауту.
  - Задача фаззера — найти аргументы, приводящие к критической ситуации в тестируемой программе.
  - Может запускать нежелательные программы, такие как *mount* и *rm -rf.*

Цель работы

Сформировать набор средств сборки и тестирования приложений для контейнера в режиме безопасных вычислений на платформе “Эльбрус”.

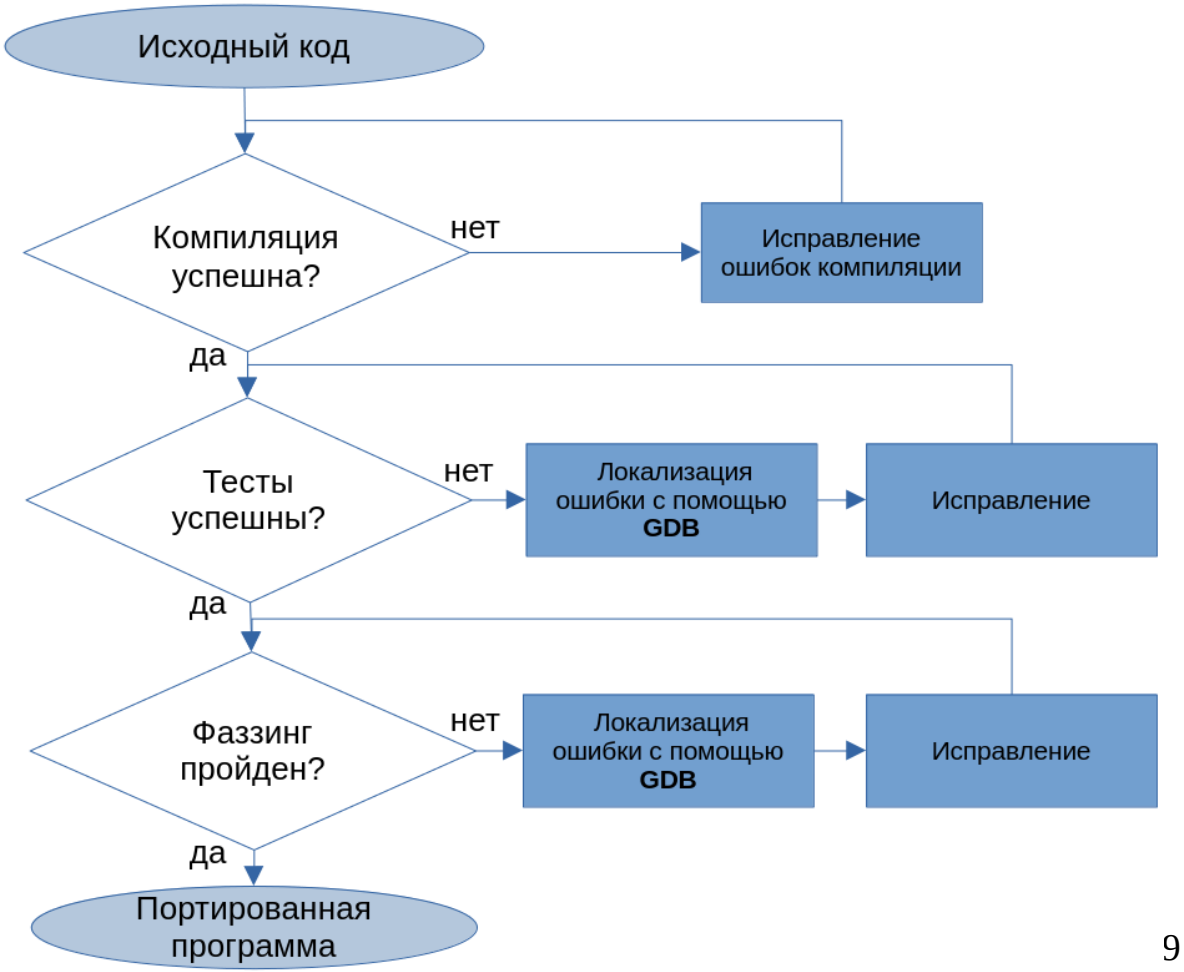
Задачи:

- Портировать в РБВ:
  - GNU make;
  - GNU Autotools и GNU M4;
  - DejaGnu, Expect и Tcl;
  - CMake.
- Портировать в РБВ и доработать iFUZZ:
  - Генерация аргументов на основе документации.
  - Учёт чёрного списка нежелательных программ.
  - Устранение недочётов, найденных в процессе работы.

Портирование программ в режим безопасных вычислений

В ходе работы портирование выполнялось в 3 этапа:

- Исправление ошибок компиляции.
- Исправление ошибок исполнения при помощи отладчика и тестов, поставляемых с программой.
- Фаззинг-тестирование программы для выявления и исправления ошибок, не покрытых тестами.



Портирование интерпретатора Tcl

- Ограничения режима безопасных вычислений:
  - Копирование структур, содержащих дескрипторы, при помощи *memcpy.*
    => Добавление присваивания новых дескрипторов старым.
  - Вывод указателя в строку в текстовом виде и последующее извлечение указателя из строки.
    => Хеш-таблица адрес-дескриптор. Запись в таблицу в месте печати адреса. Извлечение дескриптора из таблицы в месте чтения адреса.

```
// Было:
token = Tcl_CreateCommand(<аргументы>);
sprintf(buf, "%p", (void *)token);
…
if (sscanf(argv[2], "%p", &l) != 1) {
    Tcl_GetCommandFullName(interp, \
        (Tcl_Command) l, objPtr);
```

```
// Стало:
token = Tcl_CreateCommand(<аргументы>);
sprintf(buf, "%p", (void *)token);
<Таблица[buf] = token>
…
if (sscanf(argv[2], "%p", &l) != 1) {
    <l = Таблица[argv[2]]>
    Tcl_GetCommandFullName(…, (Tcl_Command) l,…);
```

- Исправлены следующие ошибки:
  - Использование неинициализированных переменных и полей структур.

Портирование процессора макросов m4

- Ограничения режима безопасных вычислений:
  - Выравнивание при помощи побитовых операций.
    => Замена побитовых операций на арифметические.

```
// Было:
char *p = (char *) ((((uintptr_t)mem + <смещение_1>) & <маска> + <смещение_2>);
// Стало:
char *p = (char *) ((((char *)mem + <смещение_1'>) + <смещение_2'>);
```

  - Преобразование типов дескриптора через *uintptr_t* для игнорирования предупреждения о рискованном приведении структуры одного типа к другому.
    => Замена на промежуточное преобразование в *char *.*

```
// Было:
argv = (token_data **) ((uintptr_t) obstack_base (&argv_stack) + argv_base);
// Стало:
argv = (token_data **) ((char *) obstack_base (&argv_stack) + argv_base);
```

Портирование системы сборки CMake

- Ограничения режима безопасных вычислений:
  - Проверка совпадения размеров *(u)intptr_t* и указателей.
    => Замена проверки на заведомо верную.
  - Сравнение *unsigned long* с *C++ nullptr* приводило к ошибке компиляции.
    => *З*амена nullptr на *NULL.*
  - Один из тестов ожидал *SEGFAULT* при обращении по неверному указателю. В РБВ это приводит к *SIGILL.*
    => Исправление *SEGFAULT* на *SIGILL* в тесте.
  - Отброс квалификатора *const* приведением типов через *uintptr_t*. Это было сделано из-за того, что библиотека *zlib* не *const*-корректна.
    => Удаление промежуточного преобразования.

```
// Было: deconst(const void *c) {
          return (void *)(uintptr_t)c;
        }
```

```
// Стало: deconst(const void *c) {
           return (void *)c;
         }
```

Портирование системы сборки CMake

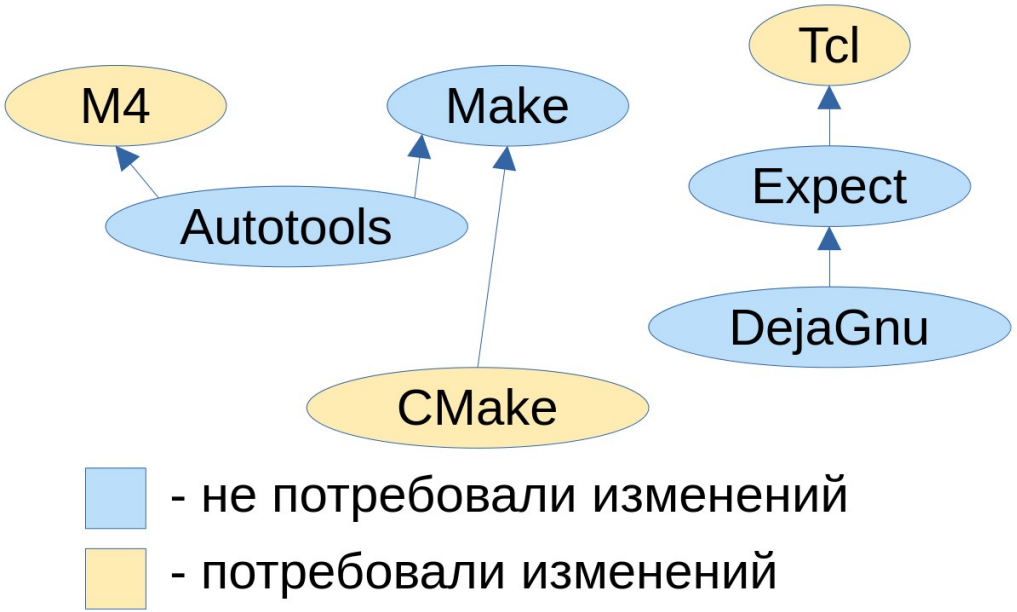
- Ограничения режима безопасных вычислений:
  - Хранение флагов цвета красно-черного дерева в последних 2-х битах указателя, подразумевая, что выравнивание будет как минимум на 4.
    => Вынос флагов в отдельную переменную.
- Исправлены ошибки:
  - Использование неинициализированных полей структур, переменных и данных, выделенных в куче.
  - Некорректное использовании функции с переменным числом параметров. Функция ожидает аргумент *long*, читая 8 байт со стека. Вызов передает *int* вместо *long*, т. е. 4 байта. Значение прочитанного аргумента не предсказуемо.

```
curl_easy_setopt(this->Curl, CURLOPT_FAILONERROR, 1); 1L);

CURLcode Curl_vsetopt(…, va_list param) {
    data->set.http_fail_on_error = (0 != va_arg(param, long)) ? TRUE : FALSE;
```

Программы, не потребовавшие изменений

- Остальные программы не потребовали внесения изменений:
  - Make
  - Autotools
  - Expect
  - DejaGnu



Доработка iFUZZ

IFUZZ не потребовал внесения изменений для работы в РБВ.

- Добавлена генерация аргументов из man-файлов документации. Разбор man-файлов заимствован из фаззера **ansvif**.
- Добавлен чёрный список запускаемых команд с поддержкой регулярных выражений в имени команды и аргументах.
- Исправлены недочёты:
  - Нет возможности работы с отдельно взятой программой.
  - Нечитаемый отладочный вывод из-за длинных генерируемых аргументов.
    => Реализовано несколько уровней подробности вывода с возможностью сокращения длинных строк.
  - Алгоритм завершения по тайм-ауту завершает тестируемый процесс, не давая ему корректно завершить порождённые процессы.
    => Процессу даётся время на обработку SIGTERM перед отправкой SIGKILL.
    => При запуске программы создаётся новая группа процессов. При завершении — завершается вся группа.

Апробация средств сборки

- Была осуществлена сборка и тестирование пакета *procps* (набор утилит для файловой системы */proc*) в контейнере РБВ с использованием портированных инструментов.
- В процессе тестирования была обнаружена и исправлена ошибка выхода за границу массива.

Результаты работы

- В режим безопасных вычислений на платформе “Эльбрус” были портированы:
  - GNU make;
  - GNU Autotools и GNU M4;
  - DejaGnu, Expect и Tcl;
  - CMake.
- Был портирован и доработан фаззер iFUZZ для работы в режиме безопасных вычислений. Доработанный фаззер назван elfuzz.
- Портированные инструменты вошли в состав контейнера РБВ.