# ОТЛАДЧИК ПАРАЛЛЕЛЬНЫХ ПРОГРАММ ДЛЯ ОС UNIX/LINUX

## РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

## ДЛЯ ВЕРСИИ 7.x

**СОДЕРЖАНИЕ**

## ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

| | | |
|---|---|---|
| HDF | – | Формат файла, содержащего очень большое количество цифровых данных. |
| GPT | – | Google performance tools, программный пакет профилирования использования программой памяти и процессора. |
| Linux | – | Unix-подобная операционная система на базе ядра Linux. |
| PD | – | Parallel Debugger, параллельный отладчик, предназначенный для отладки параллельных программ, представленный в этом документе. |
| GDB | – | Свободно распространяемый отладчик программ. |
| GDB/MI | – | GDB's Machine Interface, машинно-ориентированный текстовый протокол обмена. |
| MPI | – | Программный интерфейс передачи сообщений в параллельных программах, выполняющихся на многопроцессорных ЭВМ с распределенной памятью. |
| OpenMP | – | Программный интерфейс передачи сообщений в параллельных программах на многопроцессорных ЭВМ с общей памятью. |
| ВС | – | Вычислительная система. |
| Сопроцессор | – | Вычислительное многоядерное устройство, функционирующее внутри вычислительного узла. |
| ППМ | – | Просмотр многомерного массива. |
| СПО JAM | – | Система пакетной обработки заданий JAM. |
| SLURM | – | Система пакетной обработки заданий SLURM. |

## ВВЕДЕНИЕ

Параллельный отладчик (PD, Parallel Debugger) – масштабируемый графический отладчик, который позволяет отлаживать:

- однопроцессные и многопоточные программы;
- программы, использующие OpenMP интерфейс;
- параллельные распределенные программы с MPI интерфейсом;
- программные коды, в которых одновременно используются MPI и OpenMP интерфейсы, а также CUDA SDK (compute unified device architecture software development kit);
- клиент-серверные программы;
- гетерогенные программы, исполняющиеся на разных аппаратных платформах, в том числе и на сопроцессорах;
- программы, запускаемые как на вычислительной системе (ВС), так и на локальном компьютере в интерактивном или фоновом режиме отладки.

Параллельный отладчик позволяет осуществлять профилирование программы относительно использования оперативной памяти, эффективности использования процессора и применения MPI функций.

Параллельный отладчик поддерживает синтаксис языков программирования Фортран, Си и Си++. Возможности параллельного отладчика ограничены только аппаратурой инструментального сервера, на котором запущен его графический интерфейс, а также пропускной способностью сети, используемой для приема-передачи служебных сообщений между графическим интерфейсом и программными агентами параллельного отладчика.

Разработчики параллельного отладчика стремились сделать его графический интерфейс похожим на графический интерфейс коммерческого отладчика DDT [1]. Параллельный отладчик можно настроить на сочетание «горячих клавиш» отладчиков MS Visual Studio, Eclipse, IDEA и DDT.

Параллельный отладчик функционирует на отечественной платформе Эльбрус.

Параллельный отладчик поддерживает русский язык.

Внимание. Пробная версия параллельного отладчика обеспечивает отладку двух процессов и двух программных потоков на процесс только одним пользователем. В ней заблокирован MPI профилировщик.

В случае обнаружения ошибок, а равно как и при возникновении вопросов, предложений, сообщите о них разработчикам параллельного отладчика, отправив письмо на электронный адрес pdebugger@mail.ru

Последняя версия данного документа находится на странице https://www.pdebugger.ru

# 1. НАЗНАЧЕНИЕ И УСЛОВИЯ ВЫПОЛНЕНИЯ ПРОГРАММЫ

Назначение параллельного отладчика заключается в возможности интерактивной отладки многопроцессных и/или мультипотоковых Си/Си++, Фортран программ на ВС.

Для работы параллельного отладчика необходимы следующие программные средства:

- Java Runtime Environment версии 1.9 и выше на инструментальных серверах и вычислительных узлах ВС;
- на инструментальных серверах (локальном компьютере), вычислительных узлах GNU debugger (GDB) версии 7.x и выше. Если используется GDB без исправлений, то некоторые программные объекты в графическом интерфейсе параллельного отладчика отображаться не будут.

Отладка параллельной программы на ВС может быть выполнена, если ВС находится под управлением систем пакетной обработки заданий JAM, Open PBS или SLURM. Если применяется какая-либо другая система пакетной обработки заданий, то потребуется настроить параллельный отладчик для его взаимодействия с такой системой: создать файл с шаблоном задания, указать его в сценарии запуска графического интерфейса отладчика и т.д.

Домашний каталог отладчика параллельного отладчика размещается в доступном для всех пользователей файловой системе. Так, для отладки программ на ВС отладчик PD располагается на сетевой файловой системе, например, /nfs/COMMON/pd. Каталог pd содержит:

- bin - каталог со сценариями запуска графического интерфейса, программного агента, фонового отладчика и т.д.
- doc - каталог с документацией параллельного отладчика;
- etc - каталог с файлом политик JVM и пример файла модуля для RedHat Linux;

– lib - каталог с файлами библиотек отладчика, библиотеками взаимодействия с OpenGL/Mesa, HDF5, Google Performance Tools для Linux и т.д.
– patches - каталог с файлами программных «заплат» для исходного кода отладчика GDB;
– templates - каталог с файлами шаблонов заданий для поддерживаемых параллельным отладчиком систем пакетной обработки заданий.

## 2. ВЫБОР ОФОРМЛЕНИЯ. ЗАПУСК ПАРАЛЛЕЛЬНОГО ОТЛАДЧИКА

Параллельный отладчик позволяет выбрать цветовое оформление исходного текста программы и сочетание «горячих» клавиш (тему), используемые в отладчиках MS Visual Studio, Eclipse, IDEA, Allinea DDT (используется по умолчанию). Для установки требуемой темы необходимо инициализировать переменную окружения PD_THEME. Значение *vs* используется для темы MS Visual Studio, *eclipse* для Eclipse, *idea* для IDEA и *ddt* для Allinea DDT.

Запуск параллельного отладчика осуществляется удаленно на инструментальном сервере из терминального клиента. Если используется терминальный клиент Exceed, запущенный в ОС MS Windows, то необходимо выполнить следующие действия:

1. Войти на инструментальный сервер и в командной строке выполнить

   export DISPLAY=НАЗВАНИЕ_РАБОЧЕГО_КОМПЬЮТЕРА:0

2. Выполнить команды выборы темы (если нужно) и запуска параллельного отладчика

   export PD_THEME=vs; module load pd; pdx

Чтобы запустить параллельный отладчик из терминального клиента xterm или vncviewer, или mobaXterm [2], необходимо войти на инструментальный сервер ВС и выполнить

module load pd; pdx или $PD_HOME/bin/pdx

Для использования clipboard в vncviewer запустите утилиту vncconfig с ключом –iconic и галочкой напротив пункта, разрешающего использование данного буфера, или в настройке Options «Accept clipboard…».

## 3. ОБЩЕЕ ОПИСАНИЕ ГРАФИЧЕСКОГО ИНТЕРФЕЙСА

На рис. 1 показано графическое окно параллельного отладчика. На рисунке маркерами выделены шесть крупных графических объектов, общее описание которых приведено в таблице 1.

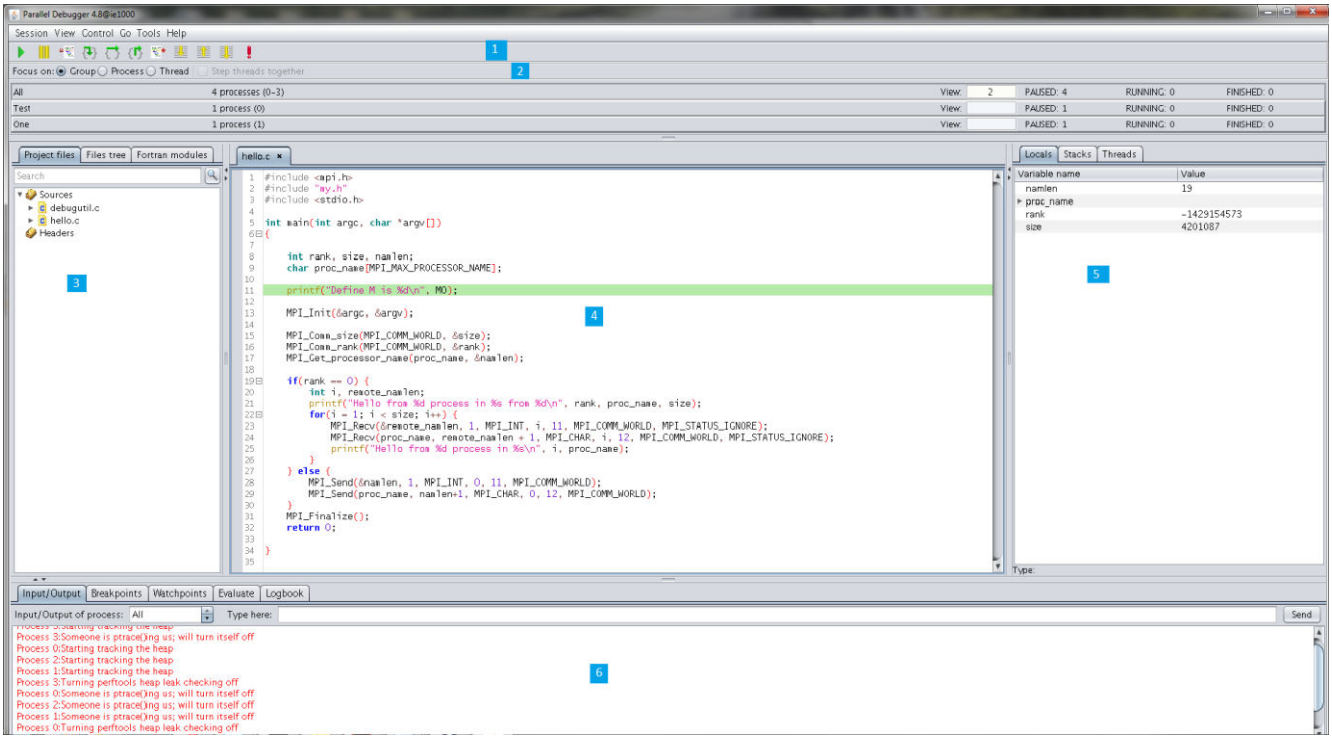



Рисунок 1 – Графическое окно отладчика. Пример сессии отладки

Таблица 1 – Список главных графических объектов отладчика

| Маркер | Описание |
|---|---|
| 1 | Панель с кнопками управления выполнением, установки точек останова, выбора кадра стека и посылки сигнала. |
| 2 | Панель выбора графического отображения номеров (рангов) отлаживаемых процессов. |
| 3 | Панель, содержащая вкладки со списком используемых в программе исходных файлов и выбора/поиска файлов на диске. |
| 4 | Панель, на которой в виде вкладок отображается содержимое отдельных |

| | |
|---|---|
| | исходных файлов отлаживаемой программы. |
|  | Панель с вкладками локальных переменных процедуры (функции) программы, кадров стека и программными потоками. |
|  | Панель с вкладками отображения стандартного вывода, точек останова, наблюдения, изменения программных переменных/выражений, журнал выполненных пользователем операций и программных событий. |

# 4. СОЗДАНИЕ ОТЛАДОЧНОЙ СЕССИИ

## 4.1 Отладка программы на вычислительной системе

Для отладки программы пользователь должен в графическом интерфейсе параллельного отладчика, используя манипулятор мышь, на инструментальной панели выбрать пункт меню «Сессия» («Session»), затем выбрать подменю «Новая сессия» («New session»), в котором для отладки на ВС щелкнуть по пункту «Конфигурирование и запуск задания» («Configure and start a job»). На рис. 2 показана последовательность действий для старта отладки программы на ВС.

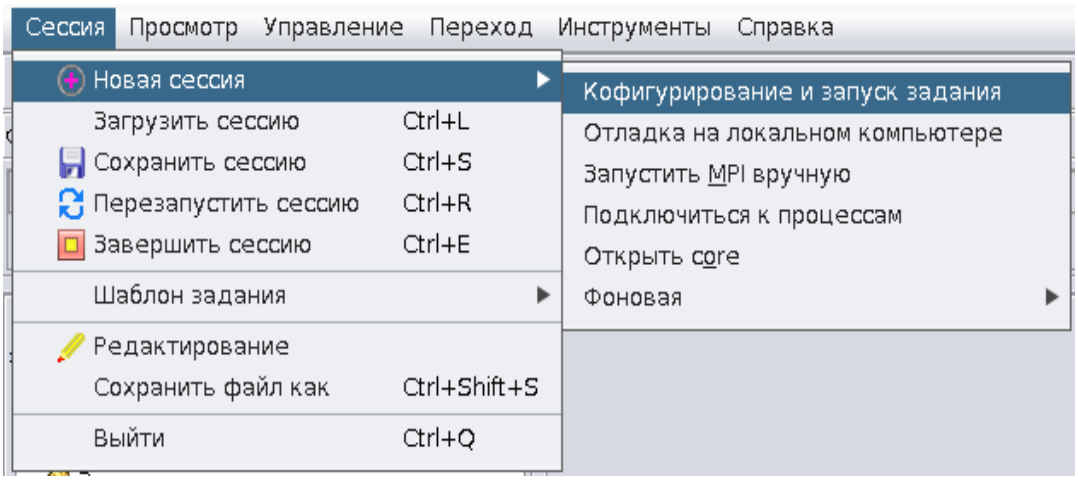



Рисунок 2 – Вызов графического окна конфигурирования задания

После этого в появившимся графическом окне (рис. 3) необходимо ввести название корневого каталога, в котором находятся подкаталоги или/и файлы с исходными текстами программы (если после сборки исполняемый файл программы не был скопирован в новый каталог, то делать это не обязательно), рабочий каталог задания, путь и название исполняемого файла.

Для упрощения ввода информации можно воспользоваться кнопками, находящимися справа от графических объектов ввода: после щелчка мышью по кнопке с иконкой появляется графическое окно, в котором в зависимости от назначения кнопки можно или выбрать каталог, или выбрать исполняемый файл. Кроме этого, в

этом же графическом окне пользователь может указать точное название метода (процедуры/функции), при вызове которого необходимо прервать выполнение программы после её запуска. Если пользователь собирается отлаживать программный комплекс, в котором задействовано несколько исполняемых файлов разных программ, то в этом случае необходимо поставить галку в графический объект с надписью «Multi-program».

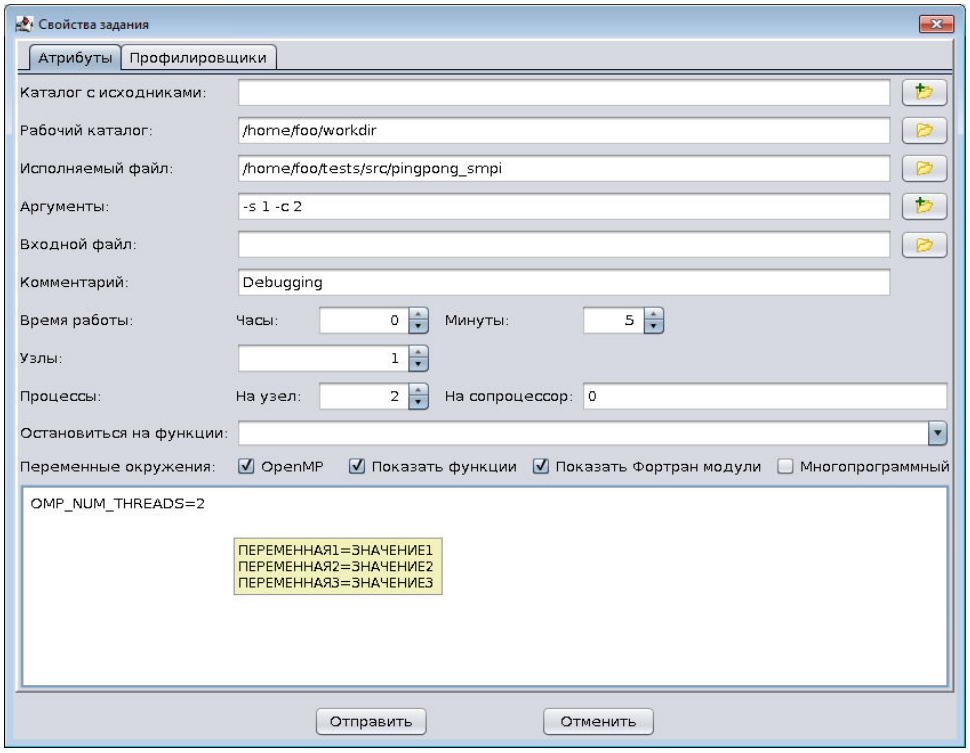



Рисунок 3 – Графическое окно ввода атрибутов задания

Передача сформированного пользователем отладочного задания системе пакетной обработки заданий осуществляется после щелчка мышью по кнопке «Отправить» («Submit»). Если на ВС нет свободных вычислительных ресурсов для запуска задания пользователя, то параллельный отладчик через некоторое непродолжительное время предложит продолжить ожидание или удалить данное задание из очереди системы пакетной обработки заданий.

Если графическое окна ввода атрибутов задания не позволяет сформировать требуемое задание, то в этом случае можно воспользоваться пунктом меню «Manual MPI

launching» (запуск MPI программы вручную).

Для отладки программы на сопроцессорах в графическом объекте «Per co-processors» пользователю надо через запятую указать количество процессов на первом устройстве (сопроцессоре), на втором и т.д. Если сопроцессоры не используются, то в данном графическом объекте должен быть указан ноль.

### 4.2 Отладка программы на локальном компьютере

Для локальной отладки (отладки непосредственно на инструментальном или рабочем компьютере) пользователь должен выбрать пункт меню «Отладка на локальном компьютере» («Debugging on the local host»). В появившемся на экране графическом окне (рис. 4) необходимо ввести путь к каталогу с исходными текстами программы (если были перемещены после сборки программы в другой каталог),

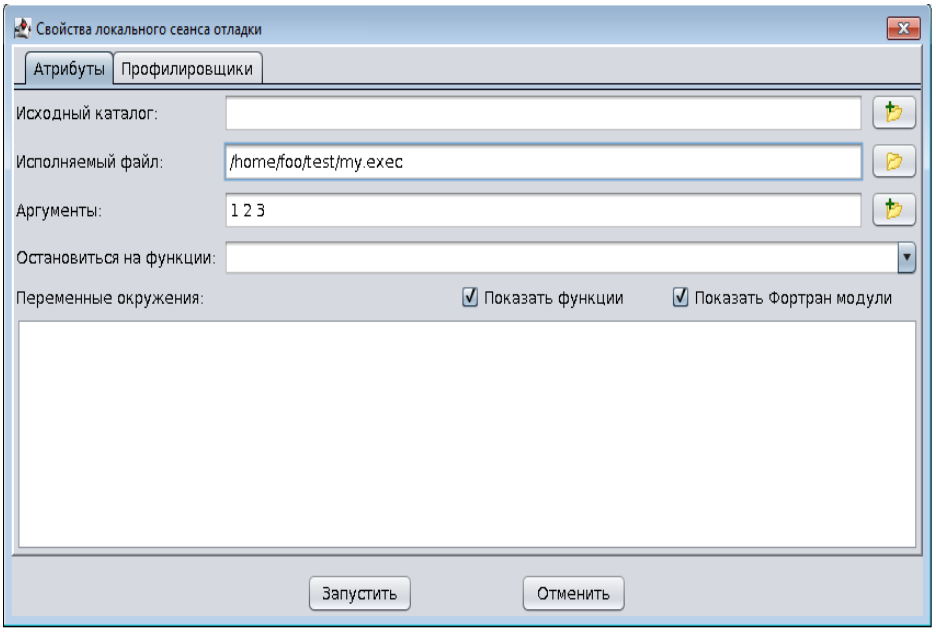



Рисунок 4 – Графическое окно формирования локальной отладочной сессии

название исполняемого файла программы и, если требуется, аргументы программы, а также название процедуры (функции) и переменные окружения. Завершив ввод, пользователь должен щелкнуть мышью по кнопке с надписью «Запустить» («Run»).

### 4.3 Отладка MPI программы, запущенной из командной строки

Для отладки MPI программы, запущенной с помощью стартера параллельных программ mpirun или mpiexec, со специфическим заказом вычислительных ресурсов, предназначен пункт меню запуска программы MPI вручную («Manual MPI launching») (рис. 2). После выбора данного пункта меню в появившемся графическом окне (рис. 5) необходимо ввести название каталога с исходными текстами программы (если каталог с исходными кодами программы был перемещен в другое место), указать (если надо) точное название используемой в программе функуции (метода), на которой надо прервать выполнение программы после её старта, а также указать общее количество процессов. Далее надо скопировать, выделив мышью, информацию из графического объекта с названием «Переменные окружения» («Use environment variable») (в качестве примера используется PD_SERVER_CONNECT=ge1001:32616). После этого щелкнуть мышью по кнопке с надписью «Ожидать» («Wait»). Отладчик перейдёт в режим ожидания подключений от программных агентов, запускаемых mpirun.

Для отладки исполняемого файла program.exec с использованием «ручного» подключения к программе графического интерфейса отладчика в командном файле/командной строке необходимо указать скопированную переменную окружения PD_SERVER_CONNECT. Пример.

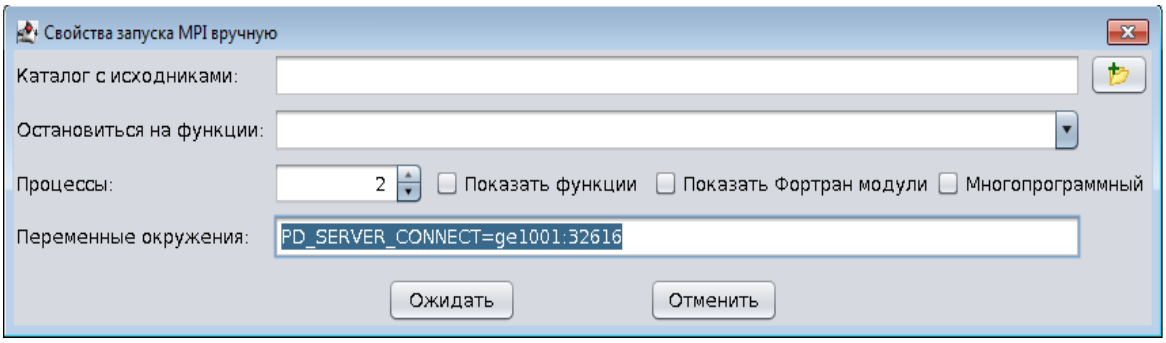



Рисунок 5 – Графическое окно формирования отладочной сессии с запуском программы пользователем

env **PD_SERVER_CONNECT=ge1001:32616** $PD_HOME/bin/agent program.exec

Для запуска MPI программы (например, 2-х процессов) требуется заранее сформировать файл (пусть его название будет HOSTS) со списком адресов компьютеров, на которых должен быть запущен исполняемый файл программы.

/opt/mvapich2-2.0/gnu/bin/mpiexec –np **2** –g **PD_SERVER_CONNECT ge1001:32616** –machinefile HOSTS **$PD_HOME/bin/agent** program.exec

Можно самостоятельно запустить (без отладчика) отладочное задание, в котором в переменных окружения указать значение PD_SERVER_CONNECT, а в директиве запуска исполняемого файла использовать $PD_HOME/bin/agent. Ниже дан пример атрибутов задания для СПО JAM.

Tasks == **2**

Exec == /usr/local/jam/maui/bin/runmpi **$PD_HOME/bin/agent** program.exec

Env == **PD_SERVER_CONNECT=ge1001:32616**

Пример атрибутов задания для SLURM дан ниже.

#SBATCH -N 1 --ntasks-per-node=2

export **PD_SERVER_CONNECT=ge1001:32616**

srun **$PD_HOME/bin/agent** program.exec

### 4.4 Подсоединение к исполняющейся программе

Подсоединиться для отладки исполняющейся программы можно с помощью последовательного выбора пунктов меню «Новая сессия» («New session») и «Подключиться к процессам» («Attach to processes») (рис. 2). После этого в появившемся графическом окне, показанном на рис. 6, следует щелкнуть мышью по кнопке «+» и ввести название сервера, на котором исполняется программа. Внимание.

На указанном компьютере уже должны быть установлены программные компоненты параллельного отладчика, а также разрешен беспарольный доступ посредством ssh (secure socket shell). Удаление выбранной строки из таблицы производится с помощью клика мышью по кнопке «-».

Для уменьшения количества отображаемой информации пользователь может ввести шаблон регулярного выражения с названием исполняемого файла программы или просто название исполняемого файла. После подтверждения ввода названия сервер на экране повится графическое окно с деревом процессов (рис. 7).

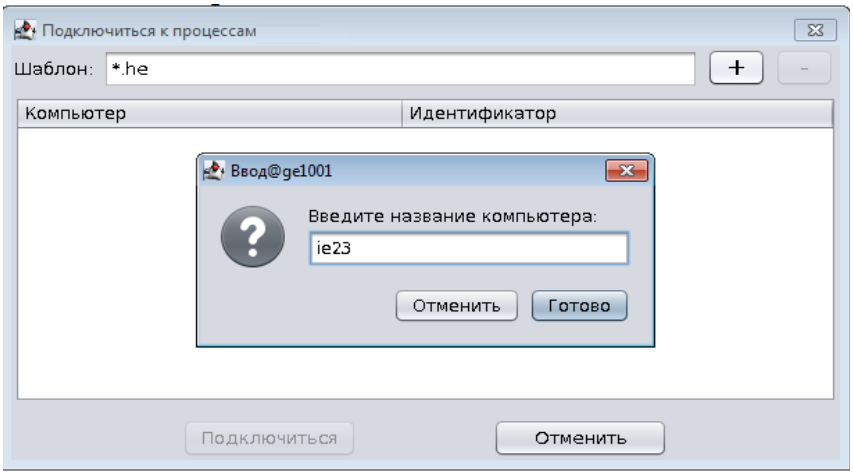



Рисунок 6 – Графическое окно формирования отладочной сессии с подключением к процессам исполняющейся программы

Для фильтрации информации использовано регулярное выражение (поле ввода «Шаблон» («Pattern») заполнено), поэтому в графическом окне на рис. 7 показана информация о процессах пользователя, исполняемые файлы которых содержат последовательность букв «he».

Выбор процессов осуществляется посредством клика мышью. Выбрать группу процессов можно с помощью мышки и нажатой клавишей Ctrl. По окончанию операции требуется подтверждение с помощью щелчка по кнопке «OK».

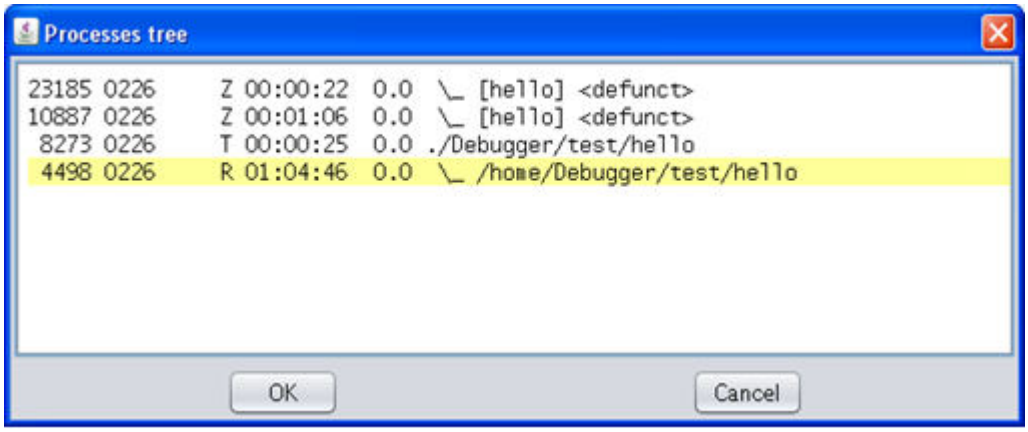



Рисунок 7 – Графическое окно выбора процесса, к которому нужно подключиться

Заполнив таблицу парами название компьютера – идентификатор процесса, пользователь должен инициировать подсоединение параллельного отладчика к процессам, щелкнув мышью по кнопке «Подсоединиться» («Attach»). В случае успешного подключения на экране дисплея отобразяться значения локальных переменных, стек, программные потоки, исходный текст программы (если доступен).

### 4.5 Отладка с использованием файлов core

Параллельный отладчик позволяет открыть один или несколько core файлов, созданных в процессе выполнения программы. Чтобы отладить программу, используя core файлы, требуется выбрать пункт меню «Новая сессия» («New session»), затем пункт «Открыть core» («Open core») (рис. 2). После этого в появившемся графическом окне (рис. 8) необходимо указать название исполняемого файла программы и с помощью кнопки со знаком плюс добавить в таблицу путь к одному или нескольким core файлам. С помощью кнопки со знаком минус можно удалить из таблицы лишние записи.

После завершения ввода пользователь должен щелкнуть мышью по кнопке «Открыть» («Open»). В графическом окне параллельного отладчика появятся группы, количество которых будет равно числу указанных пользователем файлов core.

В режиме отладки с использованием core файлов пользователю недоступны кнопки управления отладкой - «Запустить/Продолжить» («Run/Continue»), «Шагнуть» («Step

into») и т.д. Тем не менее, пользователь имеет возможность просмотреть сохраненные в файлах core переменные внутри кадров стека.

Для выхода из данного режима отладки пользователь должен выбрать пункт меню «Сессия» («Session»), а затем подпункт «Завершить сессию» («End session»).

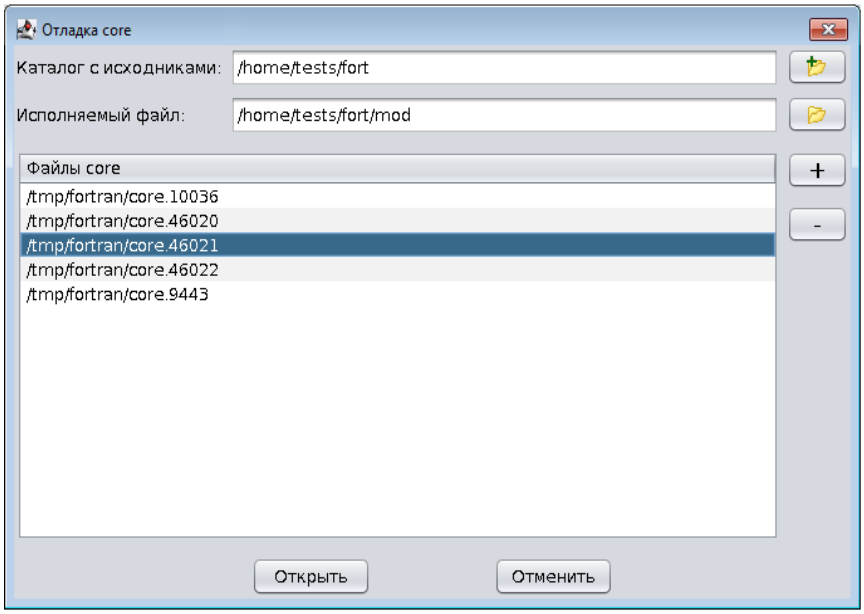



Рисунок 8 – Графическое окно выбора core файлов.

### 4.6 Отладка программы в фоновом режиме

Параллельный отладчик позволяет отладить программу в неинтерактивном или фоновом режиме (offline)[1]. Фоновый режим - это отладка без использования графического пользовательского интерфейса. Такая отладка полезна, когда, например, ресурсы ВС могут быть недоступны в течение длительного времени. Для отладки в фоновом режиме, требуется выбрать пункт меню «Новая сессия» («New session»), а затем пункт «Фоновая» («Offline») (рис. 2). После этого необходимо выбрать отладку на ВС или локальном компьютере. На рис. 9 показан графический интерфейс

[1] В пробной версии параллельного отладчика фоновый режим отладки недоступен.

формирования фонового отладочного задания. В нем пользователь должен указать атрибуты задания, а в таблицу на вкладке «Фоновая» («Offline») ввести команды, которые должен выполнить фоновый параллельный отладчик.

Команды для фонового отладчика вводятся с помощью кнопок с иконками , и надписью «Оценить» («Evaluate»), а также с помощью загрузки их из файла с расширением session. Загрузка команд из файла выполняется с помощью щелчка мышью по кнопке с изображением каталога, которая располагается справа от таблицы команд. Удалить команды из таблицы можно с помощью кнопки с изображением знака минус.

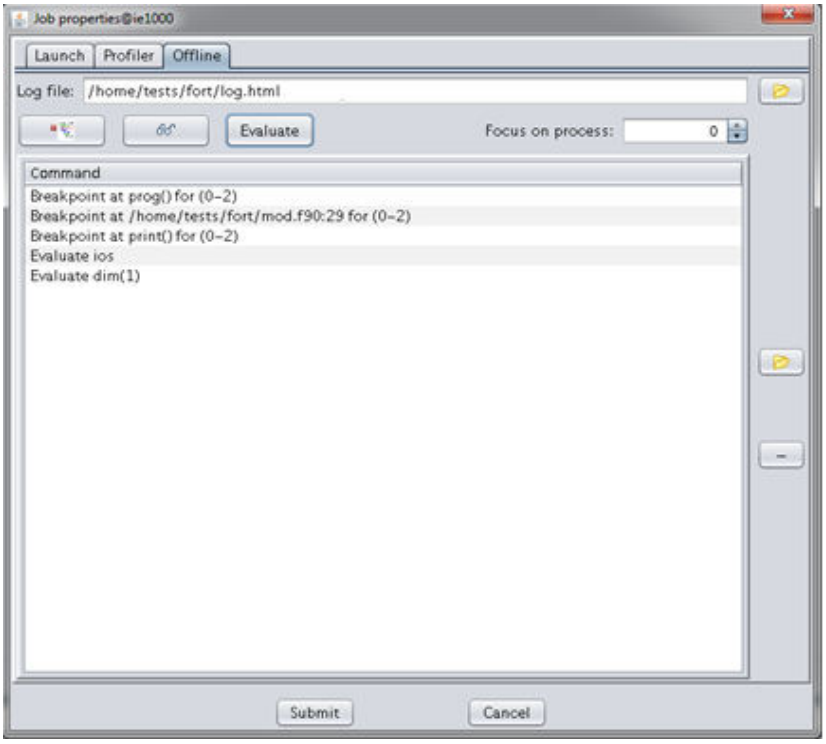



Рисунок 9 – Графическое окно создания фоновой сессии отладки

При срабатывании точки останова или наблюдения в выполняющейся программе параллельный отладчик будет считывать значения локальных переменных, используемых в процедуре/функции, данные о стеке, потоках и контролируемых глобальных переменных процесса, который был указан пользователем в поле ввода «Сфокусироваться на процессе» («Focus on process»). Перечисленную информацию

параллельный отладчик записывает в журнальный файл формата HTML, название которого пользователь должен указать на вкладке «Фоновая» («Offline») в поле «Файл журнала» («Log file»). После записи данных параллельный отладчик автоматически продолжает выполнение программы.

Информация о точке наблюдения будет записана в журнальный файл, если она сработала в контролируемом процессе. Срабатывание точки наблюдения в других процессах игнорируется. Одновременнно с информацией о срабатывании точек останова/наблюдения в контролируемом процессе в журнальный файл записывается также информация об останове других процессов, если они остановились в некотором коротком промежутке времени вместе с контролируемым процессом.

Данные профилирования программы записываются параллельным отладчиком в журнальный файл фоновой сессии в раздел «Профиль» («Profile»).

В процессе выполнения программы параллельный отладчик записывает информацию из стандартного вывода процессов программы во временный файл, который имеет название, образованное от названия журнального файла и добавления к нему расширения output. После завершения программы временный файл стандартной выдачи удаляется, а его содержимое записывается в журнальный файл фоновой сессии в раздел «Вывод» («Output»).

## 5. ИСХОДНЫЕ ФАЙЛЫ ПРОГРАММЫ. ПОИСК ФАЙЛОВ

После старта отладочной сессии названия исходных файлов программы отображаются во вкладке «Файлы проекта» («Project files»). Принимая во внимание расширения файлов, отладчик отображает их названия (без абсолютного пути) в двух группах - в группе файлов с исходным текстом программы и в группе заголовочных файлов. Если удерживать курсор мыши на назывании файла, то отображается путь к файлу. После двойного щелчка мышью по выбранному названию файла содержимое файла отображается в отдельной вкладке на панели с исходным текстом в центре графического окна отладчика (рис. 1, маркер 4).

Поиск файлов в файловой системе обеспечивает вкладка «Дерево файлов» («Files tree»). Порядок поиска файлов следующий. Сначала пользователь должен открыть каталог, в котором требуется найти файл, ввести точное название файла или шаблон поиска (регулярное выражение), а затем мышью щелкнуть по кнопке с изображением увеличительного стекла (запустить поиск).

## 6. ПАНЕЛЬ С ИСХОДНЫМ ТЕКСТОМ ПРОГРАММЫ

На рис. 10 показан пример вкладок с исходным текстом файлов программы.

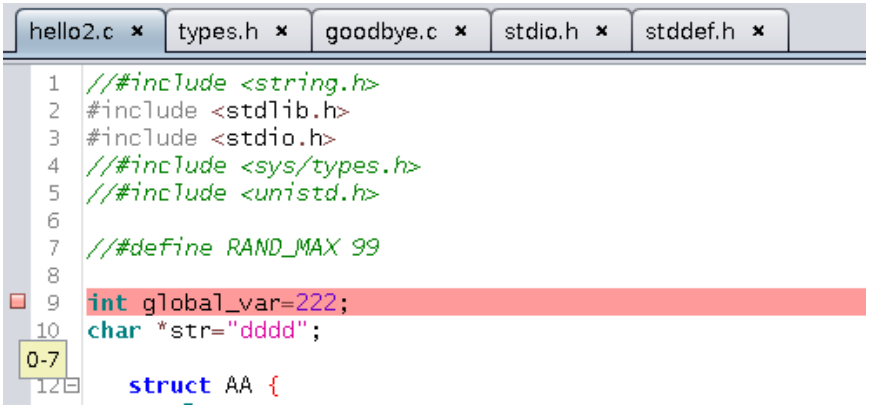



Рисунок 10 – Пример панели с вкладками исходного текста программы

На переплете вкладки с содержимым файла hello2.c расположен вертикальный ряд цифр – это номера исходных строк файла. Строка номер 9, выделенная красным цветом, содержит точку останова (прерывания). Строка, которая будет выполнена на следующем шаге группой/процессами/программным потоком выделяется зеленым цветом. Желтым цветом выделяется строка, на которой находится курсор мыши.

Красный квадратик на переплете – это иконка точки останова, которая связана с данной строкой. Всплывающая подсказка под иконкой сообщает о том, что на данной позиции должны прервать своё выполнение процессы 0-7. Если точка останова не связана с указанным процессом или группой процессов, то она не подсвечивается красной строкой, а квадрат иконки бледный, тусклый.

На каждой вкладке с содержимым исходного файла пользователь с помощью щелчка по правой кнопке мыши может вызвать всплывающее меню (рис. 11).

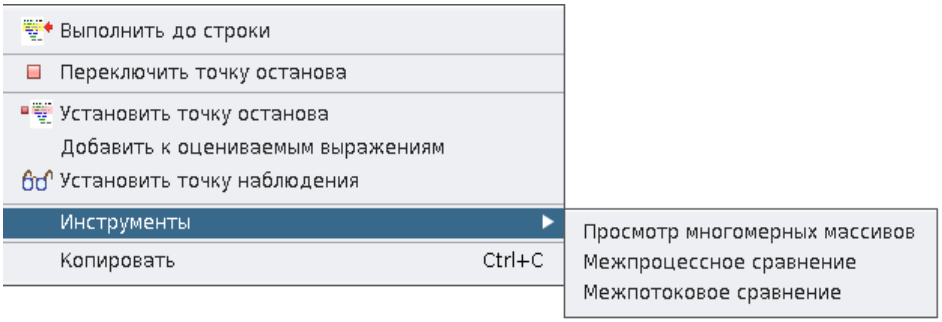



Рисунок 11 – Всплывающее меню на панели с исходным текстом

Пункты всплывающего меню позволяют выполнить следующие действия:

- «Run to here» - выполнить программу до выделенной желтым цветом строки.
- «Toggle a breakpoint» - выполнить установку/удаление точки останова в выбранных группах/процессах/программном потоке.
- «Set a breakpoint» - показать на экране графическое окно, в котором можно связать точку останова с отдельной группой/процессом/потоком, с программной процедурой (функцией), указать количество срабатываний и т.д.
- «Add to evaluations» - поместить предварительно выделенное курсором мыши выражение или переменную в таблицу на вкладке «Оценить выражения» («Evaluate»), на которой находятся оцениваемые глобальные или локальные переменные программы.
- «Set a watchpoint» - поместить предварительно выделенное курсором мыши выражение или переменную в таблицу на вкладке «Точки наблюдения» («Watchpoints»), которая содержит информацию обо всех точках наблюдения.
- «Tools» - подменю инструментов: вызов графических окон, которые позволяют сравнить программные переменные в процессах или программных потоках.

- «Copy» - копировать предварительно выделенный курсором мыши текст в буфер обмена.

Кроме выше перечисленного на вкладке реализован поиск информации в исходном тексте программы. Панель с графическими объектами, обеспечивающими поиск, появляется на экране дисплея после нажатия комбинации клавиш «Ctrl» и «F». Пример поиска в тексте программы слова «complex» показан на рис. 12.

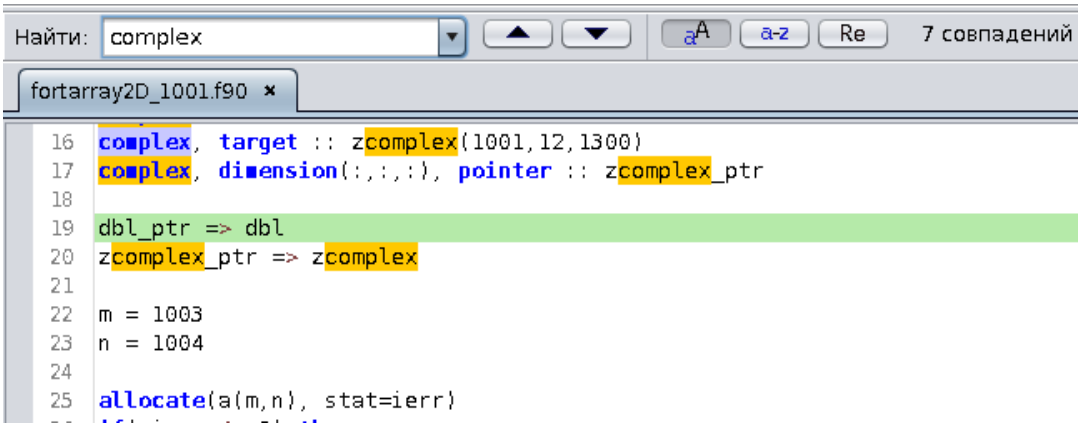



Рисунок 12 – Пример поиска слова complex в исходном тексте программы

Найденное в тексте искомое слово выделяется желтым цветом. Переход к предыдущей или следующей позиции с искомым словом выполняется после щелчка мышью по кнопкам с изображением черных треугольников. Справа от данных кнопок находятся три кнопки настройки поиска. Например, выбранная на рис. 12 кнопка с надписью «aA» обеспечивает поиск с учетом регистра.

Панель поиска можно закрыть, щелкнув на крестик в её верхнем правом углу, либо нажав на клавишу «Esc», когда курсор находится внутри графического объекта с текстом, который требуется найти.

Выделив мышью выражение и удерживая курсор мыши над выделенным фрагментом, а также удерживая курсор мыши над названием переменной, можно вызвать подсказку, в которой отобразится значение переменной и её тип (рис. 13).

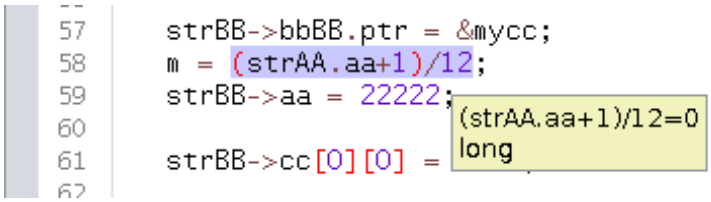



Рисунок 13 – Пример подсказки со значением выделенного в исходном тексте программы выражения

Для смены кодировки исходного текста программы, отображаемого в графическом интерфейсе параллельного отладчика, необходимо выбрать пункты меню «Просмотр» («View») и «Просмотр кода» («Code viewer»). В появившемся графическом окне выбрать название кодировки и нажать на кнопку «Применить» («Apply»). Если данные действия выполнены в режиме отладки, то изменения будут видны в следующей отладочной сессии.

Данные о кодировке, размере шрифта и табуляции записываются в файл с расширением session в каталоге $HOME/.pdx.

Данные о точках останова и наблюдения, атрибутах задания также записываются в соответствующие файлы в каталог $HOME/.pdx.

# 7. УПРАВЛЕНИЕ ОТЛАДКОЙ ПРОГРАММЫ

Пользователь может управлять процессом отладки программы с помощью кнопок управления, показанных на рисунке ниже, или применяя различные комбинации клавиш.

Кнопки управления отладкой (рис. 14) располагаются на панели инструментов в верхней части графического окна параллельного отладчика. Действия, которые выполняет параллельный отладчик после щелчка мышью по показанным на рисунке кнопками, дублируются функциональными клавишами.

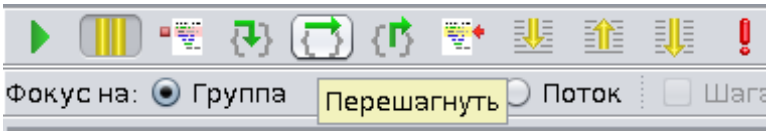



Рисунок 14 – Кнопки управления отладкой. Пример всплывающей подсказки

В таблице 2 описано назначение кнопок управления.

Таблица 2 – Описание кнопок управления (назначение клавиш соответствует Allinea DDT)

| Иконка | Описание |
|---|---|
|  | Кнопка старта/продолжения выполнения программы. Дублируется клавишей «F9». |
|  | Кнопка - индикатор останова группы/процесса/потока программы, если кнопка не утоплена (группа/процесс/поток выполняется), то по нажатию на данную кнопку процессу (процессам) посредством GDB посылается сигнал о прерывании выполнения программы. Дублируется клавишей «F10». |
|  | Кнопка вызова диалога установки точки останова (breakpoint). |
|  | Пошаговое выполнение программы, проход «внутрь» процедуры (функции) программы. Дублируется клавишей «F5». |

| | |
|---|---|
|  | Пошаговое выполнение программы без захода «внутрь» процедуры (функции) программы. Дублируется клавишей «F8». |
|  | Выполнение программы до выхода из процедуры (функции) программы. Дублируется клавишей «F6». |
|  | Выполнение программы до строки в исходном тексте, выделенной желтым цветом. |
|  | Перемещение указателя стека вниз. Дублируется последовательностью одновременно нажатых клавиш «Ctrl», «Shift» и «D». |
|  | Перемещение указателя стека вверх. Дублируется последовательностью одновременно нажатых клавиш «Ctrl», «Shift» и «U». |
|  | Перемещение указателя стека в самый низ списка кадров стека. Дублируется последовательностью одновременно нажатых клавиш «Ctrl», «Shift» и «B». |
|  | Вызов графического окна, с помощью которого можно послать процессам программный сигнал. |

После начала сессии отладки все процессы отлаживаемой программы запущены, но находятся в состоянии «приостановленные», чтобы продолжить выполнение программы требуется нажать кнопку с иконкой  . Остановить выполнение программы можно с помощью кнопки с иконкой  .

В режиме «Шагнуть» («Step into») (кнопка с иконкой  ) отладчик пошагово выполняет программу, продвигаясь по строкам исходного текста, выполняя и отображая текст процедур/функций.

В режиме «Перешагнуть» («Step over») (кнопка с иконкой  ) отладчик пошагово выполняет программу, продвигаясь по строкам исходного текста. Если в тексте программы встречается вызов функции (процедуры), то отладчик после выполнения строки, на которой находится вызов, переходит к следующей за вызовом строке.

В режиме «Вы йти» («Step out») (кнопка с иконкой ) отладчик выполняет оставшиеся до конца процедуры (функции) строки исходного текста и останавливается на следующей строке за вызовом данной процедуры/функции.

Кнопка «выполнить до указанной строки» (иконка ) позволяет выполнить программу до выбранной пользователем строки, если код относится к текущему кадру стека. Например, если пользователь выбрал строку в другой процедуре/ функции и нажал на эту кнопку, то отладчик GDB не остановит программу.

Все описанные выше кнопки действуют на выбранную пользователем группу/процессы/поток.

### 7.1 Выбор режима отладки

В параллельном отладчике реализована возможность изменения режима отладки. На рис. 15 показан графический объект, с помощью которого можно изменить режим отладки. Так, пользователь может выбрать режим «Группа» («Group») для отладки нескольких процессов одновременно. Для отладки отдельного процесса или нескольких процессов следует выбрать режим «Процесс» («Process»). Осуществлять отладку отдельного программного потока можно в режиме «Поток» («Thread»). Изменение режима приводит к автоматическому изменению информации во вкладках и таблицах графического интерфейса параллельного отладчика.


Рисунок 15 – Графические объекты смены режима отладки и отображения информации

#### 7.1.1 Режим «Group»

В режиме «Группа» («Group») можно выполнять отладку процессов программы в соответствии с их функциональным назначением: в этом режиме можно разделить

процессы параллельной программы на группы, а группы назвать в соответствии с выполняемыми ими операциями. На рис. 16 показан пример отображения группы процессов «Все».

Переключение на режим «Группа» выполняется щелчком мышью на графическом объекте с надписью «Group» (рис. 15). После этого происходит смена отображения процессов, на экране появляется графический объект с надписью «Все» («All»), который всегда находится на показанной на рисунке графической панели. Группу «Все» нельзя ни удалить, ни модифицировать. Она создаются автоматически после старта отладочной сессии.

Для того чтобы узнать ранги процессов группы, которые находятся в состоянии «приостановленные», «выполняющиеся» или «завершившиеся»,

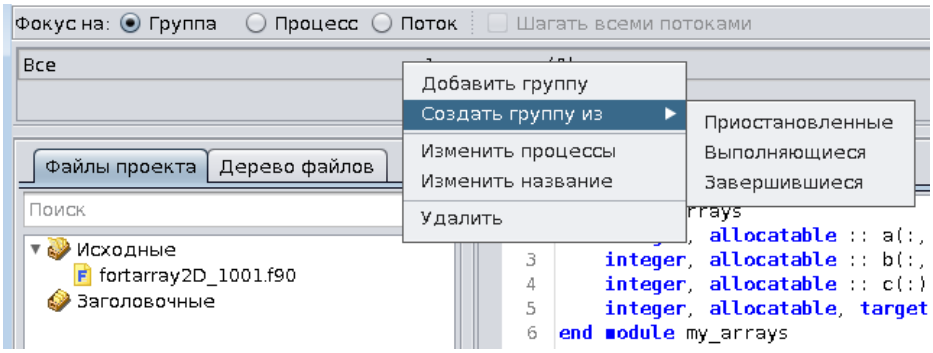



Рисунок 16 – Пример отображения группы процессов «Все»

надо подвести указатель мыши к надписи «Приостановленные» («Paused»), «Выполняющиеся» («Running») или «Завершившиеся» («Finished»), после чего на экране отобразится список рангов процессов.

Чтобы выделить процессы, находящиеся в состоянии «приостановленные», «выполняющиеся» или «завершившшившиеся» в отдельную группу, надо щелкнуть правой кнопкой мыши по панели с группами, а затем выбрать пункт меню «Создать группу из» («Create group from»).

У уже сформированных пользователем групп процессов можно изменить номера процессов, а также названия групп.

Для удаления группы необходимо навести на нее курсор мыши, а затем, щелкнуть по ней правой кнопкой мыши, вызвав всплывающее меню, в котором выбрать пункт «Удалить» («Delete»).

На графическом объекте, изображающем группу процессов, поле ввода с надписью «Наблюдать» («View») служит для отображения программных пременных, списка кадров стека и номеров потоков только этого процесса (невозможно показать всю информацию всех процессов группы одновременно). Ввод номера процесса в это поле надо завершать нажатием на клавишу «Enter».

При изменении значения программной переменной важно помнить, что изменения происходят *во всех процессах группы*, которые находятся в состоянии «приостановленные» («Paused»). Для изменения значения переменной только у процесса, номер которого указан в поле «Наблюдать», необходимо переключиться в режим отображения «Процесс» («Process»), ввести номер процесса, а потом изменить значение переменной.

### 7.1.2 Режим «Process»

Переключение на режим «Процесс» выполняется посредством щелчка мышью по графическому объекту «Процесс» («Process») (рис. 15). В поле ввода через запятую нужно указать ранги процессов или интервал рангов, используя дефис. Пример перечисления рангов процессов показан на рис. 17. Если пользователь изменил набор рангов процессов, то для переключения


Рисунок 17 – Пример режима «Process»

параллельного отладчика на новую группу необходимо нажать клавишу «Enter» или на кнопку «Готово» («Ok»). На рис. 17 показана подсказка со списком рангов процессов, которые находятся в состоянии «Приостановленные» («Paused»). Подсказка также появляется, если навести указатель мыши на надписи «Выполняющиеся» («Running») или «Завершившиеся» («Finished»).

В графическое поле ввода с надписью «Наблюдать» («View») можно ввести ранг процесса, информацию которого следует отображать во вкладках «Переменные» и «Стек» («Locals» и «Stacks») и т.д.

Внимание. Изменение значения программной переменной происходит во всех указанных пользователем процессах, которые находятся в состоянии «Приостановленные» («Paused»).

Галка в графическом объекте с надписью «Шагать всеми потоками» («Step threads together») заставляет отладчик GDB выполнять программные потоки отлаживаемой программы одновременно: при пошаговой отладке программы, в которой создаются программные потоки, отладчик будет останавливаться на первой строке исходного текста, которая должна будет выполнена потоком программы после его создания. По умолчанию в графическом объекте «Шагать всеми потоками» нет галки.

### 7.1.3 Режим «Thread»

Режим «Поток» («Thread») предназначен для отладки отдельного программного потока. После щелчка мышью по графическому объекту с названием «Поток» появляются графические объекты, в которых пользователь должен выбрать ранг процесса и номер программного потока (рис. 18).

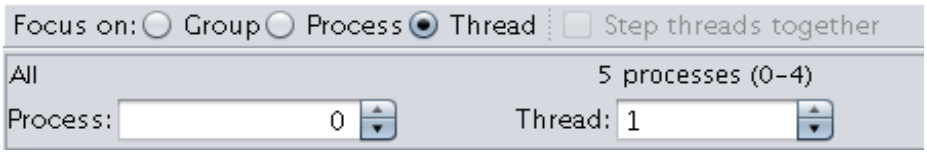



Рисунок 18 – Представление информации в режиме «Поток»

На рис. 18 показан пример, в котором выбран нулевой ранг и первый (главный) программный поток, который создаёт операционная система после запуска программы (отладчик GDB нумерует программные потоки с единицы).

### 7.2 Точки прерывания выполнения

В процессе установки точки останова (прерывания) учитывается режим выполнения: если выбран режим «Группа» («Group»), то точка останова устанавливается во всех процессах выбранной группы; если активен режим «Процесс» («Process»), то в указанных пользователем процессах, а если «Поток» («Thread»), то она устанавливается в выбранном процессе и выбранном потоке.

Независимо от выбранного режима на переплете вкладок с исходным текстом программы красным всегда подсвечиваются точки останова, действующие в процессах и потоке. Точки останова, которые установлены в других процессах или потоках, выделяются на переплете иконкой с бледно-розовым цветом. Выключенные точки останова на переплете не отображаются.

### 7.3 Точки наблюдения

При установке точки наблюдения учитываются процессы и программные потоки, указанные пользователем в выбранном режиме отладки.

### 7.4 Стек

Информация во вкладке «Стек» меняется при переключении режима отладки. В режимах «Группа» и «Процесс» («Group» и «Process») информация зависит от значения, указанного в графическом поле «Наблюдать» («View»). В режиме «Поток» отображаются локальные переменные выбранного программного потока.

### 7.5 Старт и пошаговое выполнение программы

В режимах «Группа» и «Процесс» («Group» и «Process») при старте, продолжении или пошаговом выполнении будут запущены указанные процессы (рис. 16). В режиме «Поток» («Thread») будет выполняться выбранный пользователем программный поток.

### 7.6 Настройка обработки сигналов

Пользователь может изменить обработку сигналов отладчиком GDB с помощью графического окна «Обработка сигналов» («Signal handling») (рис. 19), данное графическре окно можно вызвать с помощью пункта меню «Управление» («Control»).

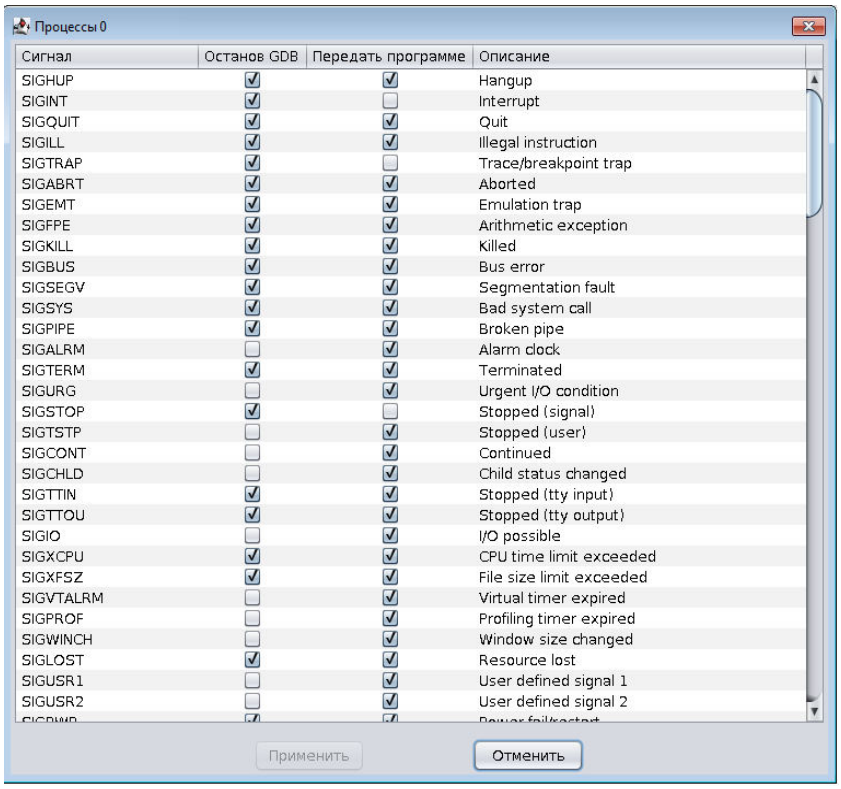



Рисунок 19 – Графическое окно изменения обработки сигналов

Установленная галка в столбце «Останов GDB» («GDB stop»), вынудит GDB остановиться при получении данного сигнала, а сняв или поставив её в столбце «Передать программе» («Pass to program») можно подтвердить или запретить передавать сигнал отлаживаемой программе.

Обработка сигналов изменяется одновременно в отладчиках GDB выбранной группы процессов.

Внимание. Не изменяйте обработку сигнала SIGSTOP, он используется параллельным отладчиком.

### 7.7 Отправка сигналов

Для отправки программного сигнала процессу пользователь должен щелкнуть мышью по кнопке с иконкой  , затем выбрать сигнал из списка, а после этого послать сигнал процессам с помощью щелчка мышью по кнопке «Послать» («Send»). Если требуется послать, например, сигнал с номером 100, то пользователю нужно выбрать в списке сигналов пункт «Другой» («Other»), а затем в появившемся графическом объекте выбрать/ввести номер этого сигнала.

Сигнал посылается всем процессам группы.

### 7.8 Запуск, завершение и перезапуск сессии отладки

Запустить, завершить или перезапустить сессию отладки пользователь может с помощью пункта меню «Сессия» («Session»).

Если отладка программы выполняется на ВС, то при перезапуске отладочной сессии процесс отладки будет выполняться в рамках уже созданного задания (новое задание создаваться не будет). Не забывайте о том, что MPI программа после перезапуска работать не будет.

При завершении или перезапуске сессии отладки информация из журнала выполненных операций и программных событий стирается.

### 7.9 Сообщения программы

#### 7.9.1 Сообщения при пошаговом выполнении

При попытке запустить программу в режиме «Группа» («Group») без выбранной группы:

«Choose at least one group» - выберите хотя бы одну группу (процессов).

В режиме выполнения «Процесс» («Process»), если указан несуществующий ранг процесса:

«The process rank "99999" does not exist» - ранга 9999 не существует.

Во всех режимах, если все указанные процессы завершились:

«All specified processes are finished» - все указанные процессы завершились.

Во всех режимах, если в дополнение к завершившимся процессам один из программных агентов также завершился, появляется сообщение о закрывшемся соединении и предложение перезапустить задание или процесс:

«Remote host closed connection. Resubmit job or launch process again».

### 7.9.2 Сообщения при завершении отладчика GDB или программного агента

При завершении отладчика GDB, который связан с некоторым процессом отлаживаемой программы, пользователю будет выдано сообщение о таком событии, например:

«Native debugger of process 0 died. Resubmit job or perform manual launching again».

В этом случае можно продолжить отладку других процессов, но отладка процесса с рангом 0 уже невозможна, поэтому параллельный отладчик рекомендует создать новое задание или запустить программу заново.

### 7.9.3 Сообщения при старте/рестарте отладочной сессии

При рестарте отладочной сессии параллельный отладчик может выдать следующие диагностические сообщения:

«The properties of the parallel debugger do not specify the paths to the preloaded profiler and unwind libraries» - в стартовом файле отладчика не указаны пути к предзагружаемым библиотекам профилировщиков, поэтому профили программы не сформируются.

«To process and analyze the profile, it is necessary to increase the job execution time by about 7 minutes» - при использовании профилировщиков необходимо увеличить время выполнения задания на указанное количество минут для сбора и анализа профиля.

«Parallel debugger agent died or native debugger has been terminated. Resubmit the job or perform manual launching again» - нельзя рестартовать отладочную сессию, так как

какой-либо отладчик GDB завершился, надо новое задание или запустить программу заново.

«Can't stop a process» - при рестарте сессии отладчик попытался остановить выполняющийся процесс, но процесс не останавливается. Рестартовать отладочную сессию нельзя. Надо или дождаться завершения процесса, или создать новую отладочную сессию.

# 8. СТАНДАРТНЫЙ ВВОД-ВЫВОД ПРОЦЕССОВ

В процессе отладки стандартный вывод процессов отображается на вкладке «Ввод/Вывод» («Input/Output») (рис. 20).

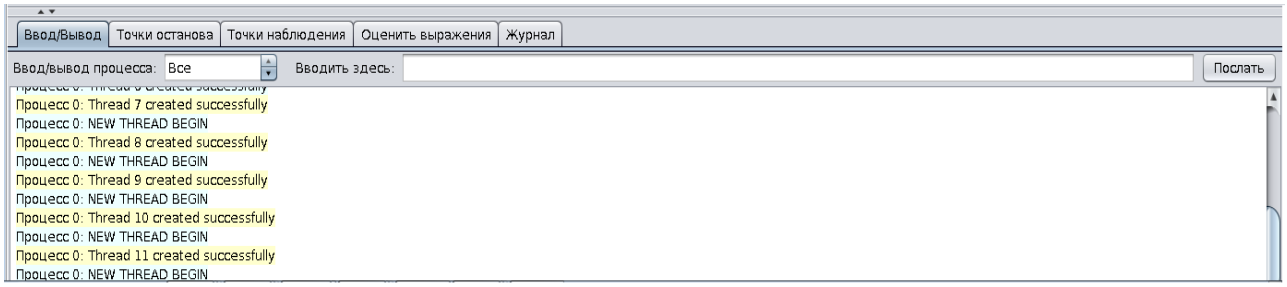



Рисунок 20 – Пример вкладки «Ввод/Вывод»

Информация из стандартного вывода и диагностики предваряется номером процесса, который её сгенерировал. Диагностическая информация выделяется красным цветом.

С помощью вкладки «Ввод/Вывод» пользователь может послать информацию на стандартный ввод группы/процесса. Передача введенной пользователем информации будет выполнена после щелчка мышью по графической кнопке с надписью «Послать» («Send»). Например, чтобы послать информацию процессу с рангом 5, пользователь должен заменить «Все» на цифру 5 в графическом объекте «Ввод/вывод процесса» («Input/Output of process»), затем ввести данные в графическом объекте ввода справа от надписи «Вводить здесь» («Type here») и нажать мышью на кнопку «Послать».

Кроме перечисленного выше пользователь может осуществить передачу информации, считанную из файла. Для этого необходимо щелкнуть правой кнопкой мыши по графическому объекту, который обеспечивает ввод информации (находится справа от надписи «Вводить здесь»). В появившемся графическом окне надо выбрать название файла. Информация из файла будет послана отлаживаемым процессам автоматически.

Если параллельный отладчик получил и отобразил информацию из стандартного вывода или диагностики выполняющихся процессов, а вкладка «Ввод/Вывод»

(«Input/Output») не выбрана, то отладчик сигнализирует об этом пользователю, добавляя к названию вкладки звездочку.

С помощью всплывающего меню, которое вызывается по щелчку правой кнопки мыши по вкладке, пользователь может:

- скопировать выделенный фрагмент страндартной выдачи или диагностики;
- выделить всю информацию;
- осуществить поиск текста в страндартной выдачи/диагностики (используется комбинация клавиш Ctrl/F);
- сохранить всю информацию или её выделенный фрагмент в файл;
- очистить графический объект, в котором отображается стандартная выдача/диагностика процессов, от информации.

# 9. ТОЧКИ ОСТАНОВА

Точка останова прерывает выполнение программы всякий раз, когда её выполнение достигает исходной строки программы, с которой она связана. Для точки останова можно добавлять условие срабатывания. Кроме этого, можно устанавливать точку останова на функцию (процедуру).

Установка, блокирование, удаление, изменение параметров точек останова может быть выполнено с помощью таблицы, пример содержимого которой показан на рис. 21.

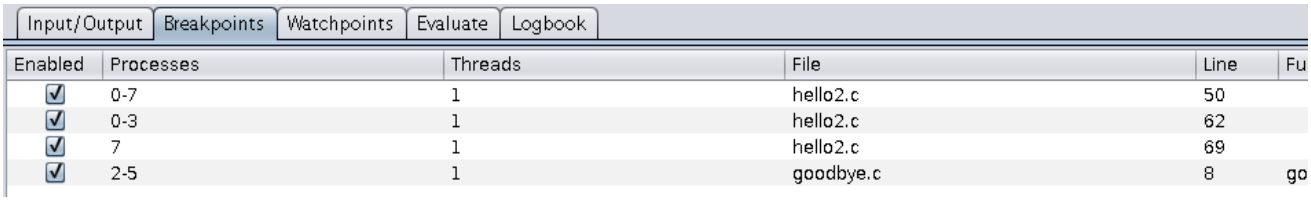



Рисунок 21 – Пример таблицы на вкладке «Breakpoints»

## 9.1 Установка точки останова. Графическое окно «Set a breakpoint»

Графическое окно «Установить точку останова» («Set a breakpoint») (рис. 22) можно вызвать с помощью щелчка мышью по кнопке с иконкой на инструментальной панели параллельного отладчика. Данное графическое окно также можно вызвать с помощью пункта «Установить точку останова» в меню на вкладке с исходном текстом программы или всплывающем меню таблицы, изображенной на рис. 21.

С помощью графического окна «Установить точку останова» пользователь может указать в какой группе/процессе/потоке[2] должна быть установлена точка останова. Кроме этого, в графическом окне можно определить при каком условии должна сработать точка останова.

Если пользователю необходимо установить точку останова на вызове функции (процедуры) программы или функции из системной библиотеки, например, malloc() или printf(), то надо поставить точку в графическом объекте «Функция» («Function») и

[2] В пробной версии параллельного отладчика току останова можно установить только в двух программных потоках.

справа от него ввести название требуемой функции (процедуры). Если пользователь пытается установить точку останова на функцию, которая не определена, то параллельный отладчик попросит подтвердить данное действие. После этого точка останова будет применена к любым объектам с указанным названием, которые будут загружены в будущем вместе с разделяемыми библиотеками.

Пользователь может указать количество пропусков или количество раз игнорирования точки останова (графический объект «Игнорировать срабатываний» («Ignore hits»)). Если требуется, чтобы после первого останова программа останавливалась в каждой итерации цикла на точке с «ignore hits» больше нуля, то «ignore hits» надо сбросить в 0, пользуясь для этого, например, таблицей, показанной на рис. 22.

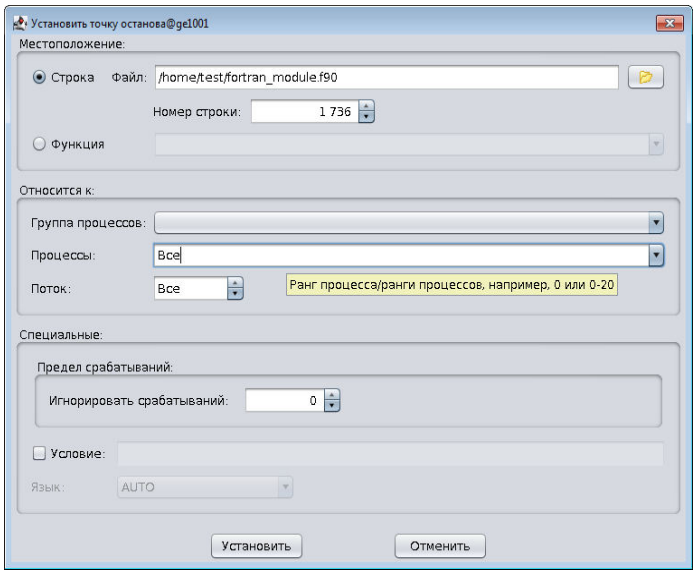



Рисунок 22 – Пример графического окна установки точки останова

Если необходимо прервать выполнение программы при достижении какого-то условия, то условие надо ввести в графический объект ввода «Условие» («Condition»). Например, пусть в программе есть некий цикл, в котором наращивается значение индексной переменной i:

```
for (i = 0; i < MAX; i++) {
```

* * *

}

Пусть надо остановиться, когда значение i станет равно 2016. Для этого ставим галочку в «Условие», а в строке с условием срабатывания указываем i == 2016 и нажимаем на кнопку с надписью «Установить» («Set»). После этого стартуем процессы и дожидаемся срабатывания этой точки останова.

Точка останова может не ставиться (или устанавливаться только у некоторых процессов), если исполняемые файлы отлаживаемой программы (multi-exec program) собраны из исходных файлов разных проектов, но с одинаковыми названиями. В этом случае пользователю рекомендуется разбить группу на отдельные подмножества процессов или устанавливать точку прерывания не на переплете окна с исходным тектом, а в графическом окне установки точки останова (рис. 22), указывая для каждого подмножества требуемый путь к исходному файлу.

### 9.2 Установка точки останова на вкладке с исходным текстом

Точку останова можно ввести с помощью щелчка мышью на переплете вкладки с исходным текстом программы (рис. 23), а также с помощью пункта всплывающего меню «Переключить точку останова» («Toggle a breakpoint»). После этого на переплете появляется иконка ■, а строка с текстом, на которую была установлена точка останова, выделяется красным цветом.

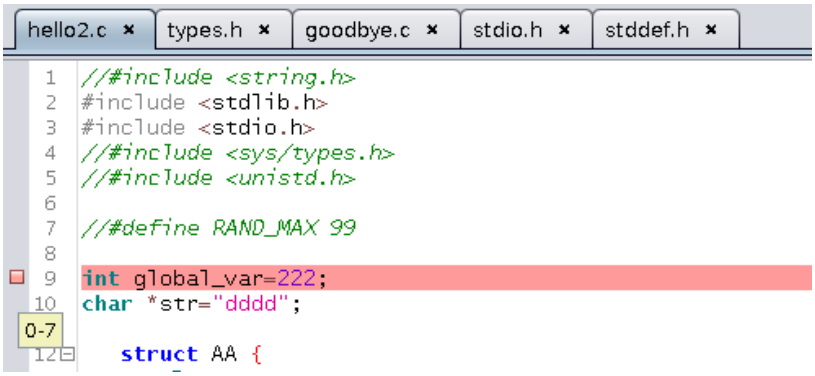


Рисунок 23 – Пример графической вкладки с исходным текстом программы

На рис. 23 красным цветом подсвечена 9 строка, с которой связана установленная точка останова. Всплывающая подсказка с цифрами «0-7» возле иконки ▫ сообщает, что данная точка действует в 0-7 процессах. При установке точки останова нужно помнить о режиме отладки. Если используется режим «Поток» («Thread»), то точка останова будет установлена в код выбранного программного потока[3].

### 9.3 Активирование/деактивирование точки останова

С помощью столбца с надписью «Включена» («Enabled») в таблице с информацией о точках останова (рис. 21) можно активировать или деактивировать точку останова. Отсутсутствие галки в ячейке столбца «Включена» говорит о том, что точка останова деактивирована.

### 9.4 Изменение информации в таблице «Breakpoints»

В таблице с информацией о точках останова доступны для изменения следующие столбцы таблицы:

- «Enabled» - если установлена галка, то точка останова активна, если галка сброшена, то точка останова деактивирована.
- «Condition» - столбец содержит условие срабатывания точки останова. Используемые в столбце операторы условия должны соответствовать языку программирования програмы.
- «Ignore» - в столбце можно изменить количество «холостых» проходов перед срабатыванием точки останова.

[3] В ознакомительной версии параллельного отладчика точка останова, установленная на переплете, действует на все программные потоки, но так как пробная версия ограничена двумя потоками, то останов нескольких потоков в режиме отладки «Поток» не обрабатывается и не отображается. Если пробная версия параллельного отладчика не выделяет зеленым цветом исходную строку в режиме отладки «Поток», то переключитесь на режим «Процесс».

Изменение информации в столбцах «Условие» («Condition») и «Игнорировать» («Ignore») должно завершаться нажатием клавиши «Enter».

### 9.5 Всплывающее меню таблицы «Breakpoints»

С помощью всплывающего меню таблицы «Точки останова» («Breakpoints») пользователь может добавить, удалить, изменить или загрузить информацию о точках останова из файла. Всплывающее меню вызывается с помощью щелчка правой кнопкой мыши по таблице. Меню содержит следующие пункты:

- «Set a breakpoint» - пункт позволяет вызвать графическое окно для установки точки останова в группе процессов, процессе или программном потоке.
- «Jump to» - выполняется переход на строку исходного файла, с которой связана точка останова.
- «Delete» - удаление выбранной точки останова из таблицы и, соответственно, удаление её в отлаживаемом процессе (процессах) программы, если они приостановлены.
- «Delete all» - удаление всех точек останова.
- «Load breakpoints» - считывание из файла и автоматическая установка точек останова. После выбора данного пункта на экране дисплея появляется графическое окно, в котором пользователь должен выбрать файл с информацией о точках останова.
- «Save breakpoints» - вызов графического окна, в котором пользователю необходимо выбрать или ввести название файла, в который будет записана информация из таблицы «Точки останова».

## 9.6 Сообщения программы

### 9.6.1 Установка точки останова

В случае неудачной попытки поставить точку останова параллельный отладчик отображает на экране сообщение с причиной, по которой нельзя установить точку останова. Например:

«Function “abrakadabra“ not defined» - функция с указанным названием не определена.

«No line 99999 in the current file» - нет строки 99999 в файле с исходным текстом программы.

«Process rank 9999 does not exist» - указан несуществующий ранг процесса.

«Invalid process rank symbol, use numbers, commas and dashes, example: 0-3,6,9-12» - использован неправильный символ в строке с рангами процессов, например, 0-3,6,9-12.

### 9.6.2 Удаление, активирование/деактивирование точки останова

"No breakpoint number 99999» - нет точки останова с номером 99999.

# 10. ТОЧКИ НАБЛЮДЕНИЯ

Точка наблюдения позволяет прервать выполнение программы, если выполнено чтение или изменение значения выражения или программной переменной.

## 10.1 Установка точки наблюдения

Установка, деактивация, удаление, изменение параметров точек наблюдения может быть выполнено в таблице «Точки наблюдения» («Watchpoints») (рис. 24).

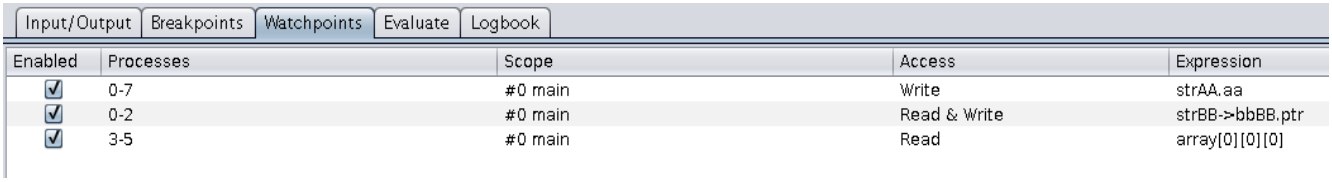



Рисунок 24 – Пример таблицы точек наблюдения

Кроме этого, установить точку наблюдения можно с помощью всплывающего меню на вкладке с исходным текстом программы (пункт «Установить точку наблюдения» («Set a watchpoint»)) и всплывающего меню вкладки «Переменные» («Locals») (пункт «Установить точку наблюдения»).

## 10.2 Активирование/деактивирование точки наблюдения

В таблице с информацией о точках наблюдения с помощью столбца «Включена» («Enabled») можно активировать или деактивировать точку наблюдения, сняв или установив галку.

## 10.3 Изменение информации в таблице «Watchpoints»

В таблице «Точки наблюдения» («Watchpoints») доступны для изменения следующие столбцы:

- «Enabled» - установленная галка активирует выбранную пользователем точку наблюдения, а если галка сброшена, то точка наблюдения игнорируется.
- «Expression» - позволяет изменить выражение или программную переменную, с которой связана точка наблюдения. Ввод нового значения в столбце «Expression» должен завершаться нажатием на клавишу «Enter».

## 10.4 Всплывающее меню таблицы «Watchpoints»

С помощью всплывающего меню таблицы «Точки наблюдения» («Watchpoints»), которое вызывается после щелчка правой кнопкой мыши по таблице, можно выполнить следующие операции:

- «Set a watchpoint» - установить точку наблюдения с помощью графического окна установки точки наблюдения в группе/процессе/потоке.
- «Delete» - удалить выбранную точку наблюдения из таблицы и, соответственно, в отлаживаемом процессе или потоке.
- «Delete all» - удалить все точки наблюдения.
- «Load watchpoints» - загрузить и автоматически установить точки наблюдения из файла.
- «Save watchpoints» - вызов графического окна, в котором пользователь должен выбрать или ввести название файла, в который должна быть записана информация из таблицы.

## 10.5 Сообщения программы

### 10.5.1 Установка точки наблюдения

В случае неудачной попытки установить точку наблюдения параллельный отладчик выводит на экран дисплея следующие сообщения.

«No symbol "Foo" in current context» - нет переменной с названием «Foo» в текущей процедуре (функции).

«A syntax error in expression, near `+`» - неверный синтаксис выражения, сообщение появляется, например, если пользователь указал в графическом окне «Добавить точку наблюдения» язык программирования, которому не соответствует введенное выражение, или если выражение содержит ошибочный символ.

### 10.5.2 Установка точки останова

В случае неудачной попытки установить точку останова параллельный отладчик выводит на экран дисплея следующие сообщения.

«Breakpoint can't be set» - точка останова не может быть установлена.

«No debug session with process #» - нет отладочной сессии с процессом номер #, сообщение появляется, когда пользователь указал ранг несуществующего процесса.

«Select at least one process» - выберите хотя бы один процесс; данное сообщение выводится на экран, если пользователь не указал ни одного ранга процесса, или ни одной группы процессов.

# 11. ИЗМЕНЕНИЕ ЗНАЧЕНИЙ ПЕРЕМЕННЫХ

Контролировать или изменять значения программных переменных, а также вычислять выражения можно с помощью таблицы на вкладке «Оценить выражения» («Evaluate») (рис. 25).

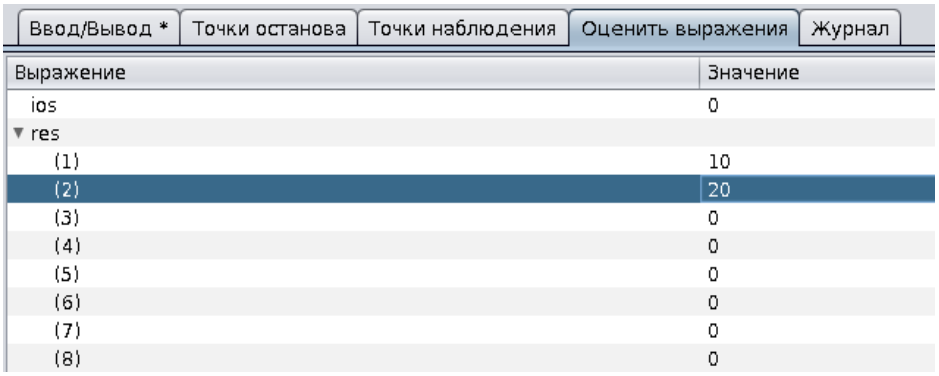



Рисунок 25 – Пример таблицы оценки/изменения переменных программы

## 11.1 Доступные операции с переменными и выражениями на вкладке «Evaluate»

### 11.1.1 Добавление

Добавить в таблицу «Оценить выражения» («Evaluate») запись о программной переменной или выражении можно с помощью пункта «Добавить к оцениваемым выражениям» («Add to evaluations») из всплывающих меню на вкладках с исходным текстом программы, «Переменные» («Locals») и «Оценить выражения» («Evaluate») (с помощью пункта «Добавить выражение для оценки» («Add expression»)). После выбора данного пункта меню введите название переменной или выражение. С помощью знака равенства можно назначить переменной новое значение. Примеры:

foo

j=1

struct%ch=’O’

При вводе выражения необходимо учитывать синтаксис используемого в

программе языка программирования (исключая указатели в Фортран программе).

На рис. 26 показан пример графического окна, с помощью которого можно ввести новое значение программной переменной или ввести выражение для оценки его значения.

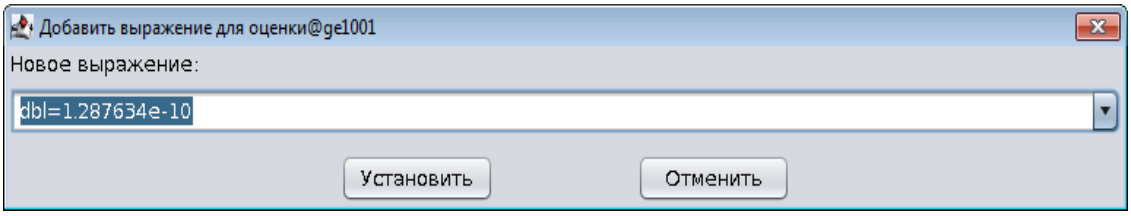



Рисунок 26 – Пример графического окно ввода значения программной переменной

### 11.1.2 Добавление указателей

Для отображения значений памяти, на которую ссылается указатель Фортран программы, указатель должен быть добавлен в виде выражения для языка программирования Си. Например, если имеется указатель, объявленный в программе как

integer, pointer :: IPTR(:)

, то его можно добавить в таблицу посредством ввода выражения *((integer *) IPTR). Кроме этого, если надо оценить только 100 значение, то необходимо использовать смещение 99, а не 100 (в языке програмирования Си индексы начинаются с 0):

*((integer *) IPTR + 99).

### 11.1.3 Редактирование выражений

Параллельный отладчик позволяет редактировать оценочные выражения (рис. 27), которые доступны для модификации. Для изменения выражения щелкните правой кнопкой мыши по таблице на вкладке «Оценить выражения» («Evaluate»), в появившемся меню выберите пункт «Изменить выражение» («Edit expression»).

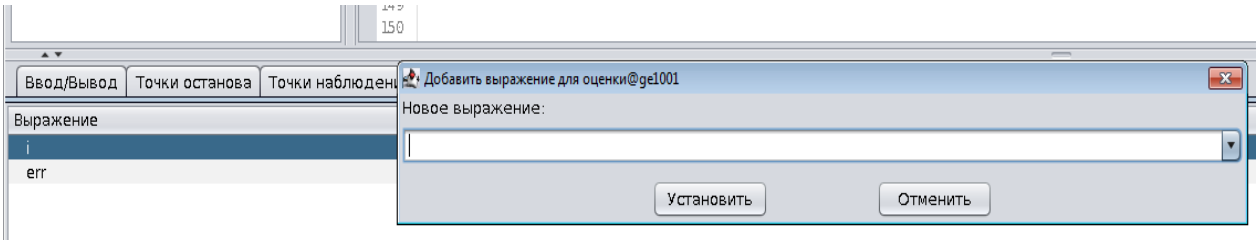



Рисунок 27 – Пример редактирования выражения с переменной local_var

После этого появится графическое окно, в котором можно отредактировать выражение. Данное графическое окно можно вызвать также с помощью двойного щелчка левой кнопкой мыши по выбранному в таблице выражению. Внимание. Вводить новое выражение следует с помощью пункта «Добавить выражение» («Add an expression»).

### 11.1.4 Изменение значений

Чтобы изменить значение переменной/выражения, щелкните по таблице «Оценить выражения» («Evaluate») правой кнопкой мыши, в появившемся меню выберите пункт «Изменить значение» («Edit value»). После этого на экране дисплея появится графическое окно, в котором следует ввести новое значение, учитывая тип переменной/выражения (если переменная – символ, то новое значение должно быть обрамлено одинарными кавычками). Данное графическое окно можно вызвать также с помощью двойного щелчка левой кнопкой мыши по выбранному в таблице значению.

## 11.2 Удаление изменений

Удалить одну строку или все строки из таблицы «Оценить выражения» («Evaluate») можно с помощью всплывающего меню этой таблицы, которое вызывается посредством щелчка правой кнопкой мыши по таблице. В появившемся меню надо выбрать пункт «Удалить» или «Удалить все» («Delete» или «Delete all»).

## 11.3 Изменение информации в таблице «Evaluate»

Изменить информацию в выбранной ячейке таблицы «Оценить выражения» («Evaluate») можно с помощью двойного щелчка мышью. После этого на экране дисплея

появляется графическое окно, в котором необходимо ввести выражение или новое значение выражения.

### 11.4 Всплывающее меню таблицы «Evaluate»

Всплывающее меню таблицы «Оценить выражения» («Evaluate») содержит следующие пункты:

- «Add expression» - вызов графического окна, в котором пользователь может указать новое значение переменной или выражения, ввести выражение для контроля значения переменной.
- «Edit expression» - вызов графического окна, в котором можно указать программную переменную или выражение, отличное от выбранного в таблице.
- «Edit value» - вызов графического окна, в котором можно ввести новое значение выбранной в таблице програмной переменной или выражения. Пункт дублирует возможность изменять значения непосредственно в столбце «Значение» («Value»).
- «Edit type/language» - вызов графического окна, в котором можно указать новый тип переменной или выражения, а также изменить язык программирования для выражения.
- «Copy value» - скопировать значение в системный буфер обмена.
- «Add to watchpoints» - добавить выбранную переменную в таблицу точек наблюдения.
- «View as» - отображение значения числовой программной переменной в десятичной, шестнадцатиричной, двоичной, восьмиричной системах счисления или экспоненциальном формате.
- «Delete» - удаление выбранной строки.
- «Delete all» - удаление всех строк из таблицы.
- «Load evaluations» - загрузка названий программных переменных или выражений из файла. После выбора данного пункта вызывается

графическое окно, в котором пользователь должен выбрать файл с информацией.

- «Save evaluations» - вызов графического окна, в котором пользователь должен выбрать или ввести название файла, в который будет записана информация из таблицы.

## 11.5 Сообщения программы

### 11.5.1 Сообщение при добавлении/модификации

«-var-create: unable to create variable object» - отладчик не может создать объект, связанный с указанной переменной или выражением. Сообщение выводится на экран дисплея, когда пользователь указал несуществующую переменную, а также если он ввел выражение с ошибкой (выражение не соответствует синтаксису языка программирования).

## 12. ЖУРНАЛ ДЕЙСТВИЙ И СОБЫТИЙ

Выполненные пользователем в процессе отладки действия, а также события, произошедшие в процессах и программных потоках, фиксируются в журнале, роль которого выполняет таблица на вкладке «Журнал» («Logbook») (рис. 28). Для привлечения внимания пользователя информация о некоторых действиях и программных событиях сопровождается специализированной иконкой.

Кроме этого, если несколько процессов программы получили какой-либо сигнал, то на экране дисплея выводится графическое окно с рангом процесса, который получил данный сигнал первым. Информация от других процессов во всплывающем графическом окне не отображается, но регистрируется в журнале событий.

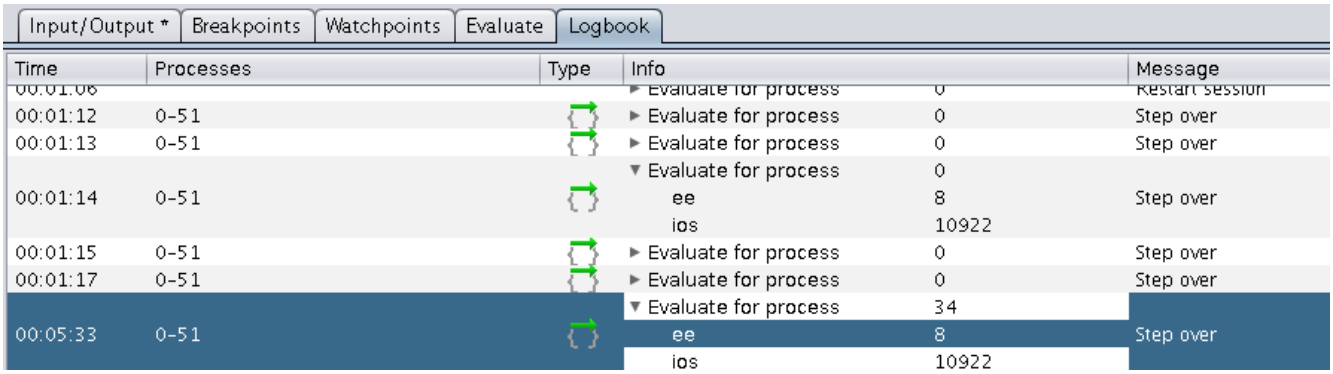



Рисунок 28 – Пример журнала действий и событий

Информацию, которая отображается в таблице на рис. 28, можно сохранить в гипертектовый файл. Для этого надо щелкнуть по таблице правой кнопкой мыши и в появившемся всплывающем меню выбрать пункт «Сохранить журнал» («Save a logbook»), а затем выбрать или указать название файла, в который надо записать информацию.

# 13. ПРОСМОТР ПЕРЕМЕННЫХ

## 13.1 Локальные переменные. Вкладка «Locals»

Вкладка «Переменные» («Locals») содержит локальные переменные наблюдаемого процесса (рис. 29). Информация в таблице отсортирована по названию программных переменных.

Внимание. На данной вкладке инициализированный Си указатель на массив не отображается как массив (отображается только его адрес в оперативной памяти). В виде массива он отображается на вкладке «Оценить выражения». Кроме этого на вкладке «Переменные» не отображается как массив инициализированный Фортран указатель на массив или allocatable переменная, если GDB не собран с программной «заплатой».

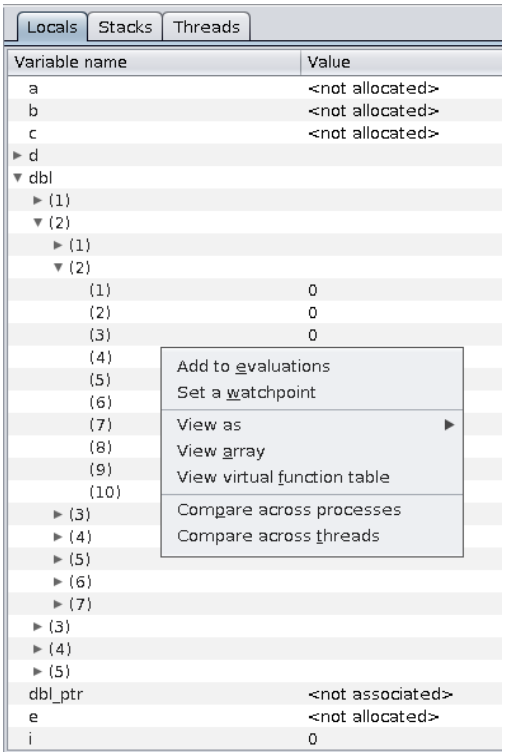



Рисунок 29 – Пример вкладки «Locals»

Всплывающее меню, показанное на рис. 29, содержит следующие пункты:

- «Add to evaluations» - данный пункт меню позволяет зарегестрировать выбранную переменную в таблице «Оценить выражения» («Evaluate»).

Внимание. После щелчка мышью, если выполняется отладка Фортран программы, может появиться графическое окно, в котором, если переменная является структурой или массивом, указан идентификатор в операторах Си. Пользователь должен исправить идентификатор сомостоятельно. Кроме этого, для Фортран программ отладчик GDB переворачивает индексы массивов, например, элемент a(6,1) будет иметь индекс a(1,6).

- «Set a watchpoint» - пункт позволяет зарегестрировать выбранную переменную в таблице «Точки наблюдения» («Watchpoints»). Внимание. После щелчка мышью, если выполняется отладка Фортран программы, может появиться графическое окно, в котором, если переменная является структурой или массивом, указан идентификатор в операторах языка Си. Пользователь должен исправить идентификатор сомостоятельно.
- «View as» - отображение значения числовой переменной в десятичной, шестнадцатиричной, двоичной, восьмиричной системах счисления или экспоненциальном формате, а также в виде массива.
- «View array» - позволяет просмотреть значения массива в специализированном графическом окне.
- «View virtual function table» - отображение таблицы виртуальных функций, связанных с программным объектом (Си++ программы), находящимся в данном кадре стека.
- «Compare across processes» - вызов графического окна, в котором можно сравнить значения выбранной переменной в нескольких процессах.
- «Compare across threads» - вызов графического окна, в котором можно сравнить значения выбранной программой переменной в программных потоках.

### 13.2 Глобальные переменные

На вкладке «Переменные» («Locals») глобальные переменные программы не отображаются, поэтому для просмотра значения глобальной переменной её название необходимо внести в таблицу на вкладке «Оценить выражения» («Evaluate»).

### 13.3 Указатели

Проинициализированные указатели можно просмотреть в виде массива с помощью обращения к пункту «Просмотреть массив» («View array») всплывающего меню на вкладке «Переменные» («Locals»). Например, пусть выполняется отладка Фортран программы:

```
integer, pointer :: IPTR(:)

nullify(IPTR)

Size=100

allocate(IPTR (Size))

do i=1,Size

   IPTR (i)=i*10

enddo
```

Если требуется просмотреть значения памяти, на которую ссылается указатель IARR, то надо обратиться к пункту меню «Просмотреть массив» на вкладке «Переменные» («Locals»). В появившемся графическом окне отобразятся размерности массива, на который ссылается IARR[4]. Если графические объекты – размерности массива - не отобразились, то, хотя исходный язык программирования Фортран, пользователю надо

[4] Отладчик GDB без исправлений не отображает проинициализированный указатель как массив, поэтому в графическом окне отобразится только одна размерность.

указать выражение для языка программирования Си: *((integer *) IPTR), где (integer *) – это приведение IPTR к указателю на тип integer. Пользователь должен указать пределы размерностей, а потом нажать на кнопку с надписью «Оценить» («Evaluate»). Заметим, что в процессе отладки программы, которая была скомпилирована GNU Fortran, указатель IPTR будет отображен на вкладке «Переменные» как массив.

Для просмотра значений, которые находятся в оперативной памяти компьютера, на которую указывает переменная-указатель Си/Си++ программы, последовательность действий аналогичная. Если в таблице значений отображаются явно неправильные данные, то необходимо привести указатель к нужному типу.

### 13.4 Кадры стека. Вкладка «Stacks»

Изменение стека фиксируется в таблице на вкладке «Стек» («Stacks») (рис. 30). Кроме этого, в строках таблицы при отладке программы отображается информация о значениях аргументов функций. Трасса выполнения (список кадров стека) при получении от операционной системы сигнала позволяет локализовать область программы, при выполнении которой произошла его генерация.

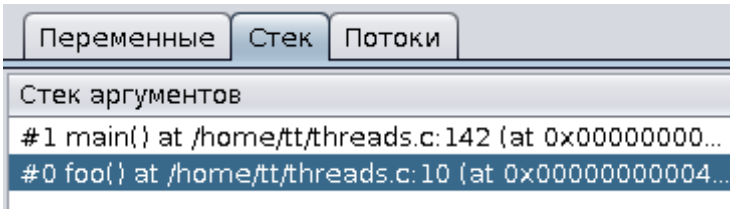



Рисунок 30 – Пример вкладки «Stacks»

В показанной на рисунке таблице можно выбрать требуемый кадр стека, щелкнув мышью по нужной строке. Одновременно с этим в исходном тексте программы будет отображена строка, связанная с выбранным кадром. Вместе с этим изменится информация на вкладках «Переменные» и «Оценить выражения» («Locals» и «Evaluate»).

### 13.5 Потоки. Вкладка «Threads»

Информация о созданных в отлаживаемом процессе программных потоках отображается в таблице на вкладке «Потоки» («Threads») (рис. 31). В таблице отображается номер программного потока, название процедуры (функции), название исходного файла и номер строки, на которой остановлено выполнение потока.

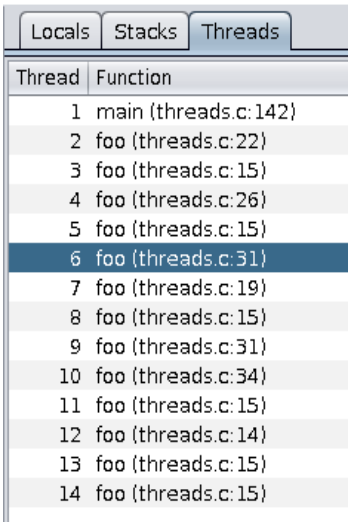



Рисунок 31 – Пример вкладки «Потоки»

С помощью таблицы «Потоки» можно выбрать нужный поток, если выбран режим отладки «Поток». Одновременно с переключением на выбранный поток изменяется информация на вкладке «Переменные» («Locals») и позиция строки исходного текста (она подсвечивается зеленым цветом[5]).

[5] Если отладчик GDB не предоставил информацию о строке в исходном тексте программы (например, выполняется код системной программной библиотеки), то параллельный отладчик не выделяет строку зеленым цветом.

## 14. ОТОБРАЖЕНИЕ ПЕРЕМЕННЫХ И ПРОЦЕДУР ФОРТРАН МОДУЛЕЙ

Реализованные в Фортран программе модули отображаются на отдельной вкладке (рис. 32), если пользователь установил галку в пункте меню «Показывать Фортран модули» («Show Fortran modules») или в графическом окне ввода атрибутов задания.

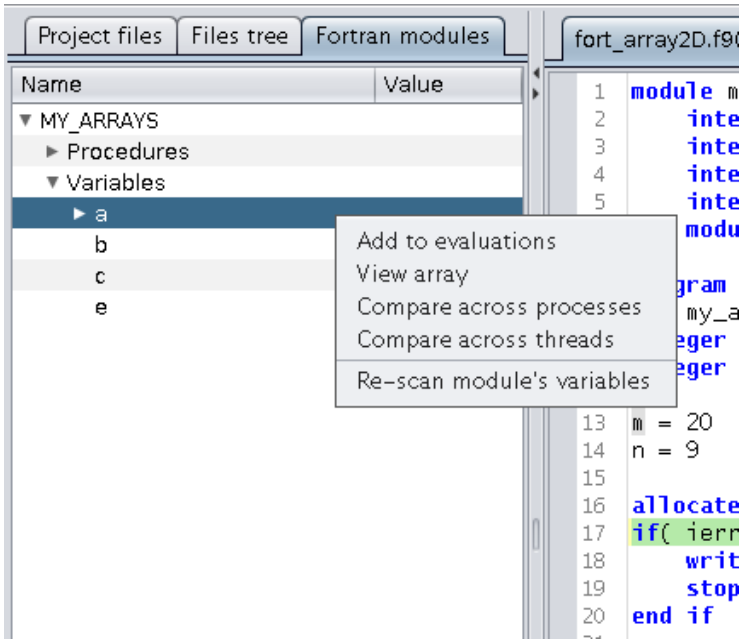



Рисунок 32 – Пример содержимого вкладки «Fortran modules»

Всплывающее меню на вкладке «Фортран модули» («Fortran modules») позволяет пользователю оперировать используемыми в модулях переменными. Содержимое данной вкладки при выполнении программы не обновляется автоматически. Для обновления информации о переменных выбранного модуля следует использовать пункт меню «Повторное сканирование переменных модуля» («Re-scan Fortran module's variables»).

Пункт всплывающего меню «Добавить к оцениваемым выражениям» («Add to evaluations») служит для изменения значения переменных в Фортран модуле. Кроме этого, изменить значение переменной Фортран модуля можно с помощью всплывающего меню вкладки «Оценить выражения» («Evaluate»). В этом случае необходимо использовать название модуля, два двоеточия и название переменной, например, MY_ARRAYS::a

Двойной щелчок мышью по названию процедуры на вкладке «Фортран модули» позволяет перейти к строке с исходным кодом данной процедуры.

Внимание. Старые версии компилятора GNU Fortran не включают в исполняемый файл отладочную информацию о программных модулях Фортран программы.

## 15. ПРОСМОТР ЗНАЧЕНИЙ ЭЛЕМЕНТОВ МАССИВА

Просмотреть значения элементов массива можно с помощью графического окна просмотра многомерного массива (ПММ) (рис. 33).

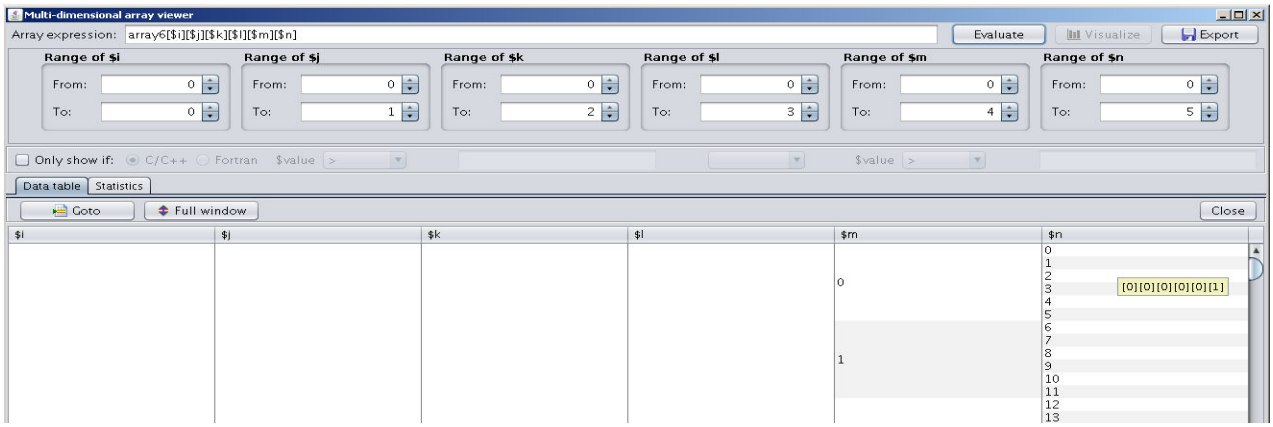



Рисунок 33 – Пример графического окна средства просмотра многомерных массивов

ПММ можно вызвать с помощью щелчка левой кнопкой мыши во всплывающих меню на вкладке с исходным текстом программы, из пункта меню «Инструменты» («Tools») или всплывающего меню вкладки «Переменные» («Locals»). В графическом окне отображаются массивы, имеющие простой тип. Массив структур, классов и так далее не отображаются.

Если ПММ вызвано с помощью щелчка мыши на названии программной переменной, которая имеет тип массив, то ПММ раскрывается с инициализированными панелями диапазонов размерностей. Если ПММ вызван, например, с помощью меню «Инструменты» («Tools»), то пользователь должен самостоятельно ввести название программной переменной. При вводе символа «[» (если программа написана на языке программирования Си/Си++) или символа «(» (для Фортрана) ПММ автоматически допишет размерности, используя мета-переменные (знак «$» и буква). В строке с названием «Выражение» («Array expression») показаны мета-переменные $i, $j, $k, $l, $m и $n. Если пользователь желает сократить область просмотра значений массива, то

он может явно указать требуемые ему индексы, например, array6[0][$i][2][3][$j][$k] для Си/Си++, а в случае Фортран программы array6(1,$i,2,3,$j,$k).

После ввода названия массива, установки диапазонов индексов («Диапазон $i» («Range of $i») и т.д.) пользователь должен щелкнуть мышью по кнопке с надписью «Оценить» («Evaluate»). ПММ начнет считывать данные, обращаясь к отладчику GDB. Прогресс считывания данных будет отображаться внизу графического окна ПММ. Если массив очень большой, то ПММ не считывает все элементы массива сразу. Считывание будет выполняться «на лету» по мере просмотра (пролистывания) значений массива в таблице.

Внимание. Элементы массива Фортран программы отображаются в порядке, который характерен для Си/Си++ программ (так их печатает отладчик GDB): элементы отображаются по строкам, а не столбцам.

### 15.1 Фильтрация значений

Если необходимо, чтобы в таблице (рис. 33) отображались цифровые значения, удовлетворяющие определенному критерию (больше или меньше нуля и т.д.), то фильтрацию информации можно осуществить с помощью установки галки в графическом объекте «Показать если только» («Only show if») и инициализации условных выражений. Например, если указано условие $value == 0 («показать все значения, которые равны нулю»), то все отличные от 0 значения элементов массива будут скрыты (здесь $value является фактическим значением отдельного элемента).

### 15.2 Переход к элементу массива

Переход к элементу массива можно выполнить с помошью графического окна, пример которого показан на рис. 34.

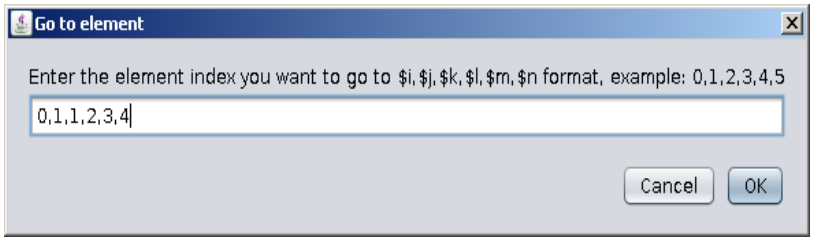



Рисунок 34 – Пример графического окна ввода индекса элемента массива

Для перехода к элементу массива необходимо щелкнуть мышью по кнопке с надписью «Перейти» («Goto»). В появившемся графическом окне «Перейти к элементу» («Go to an element») через запятую надо ввести индексы требуемого элемента. Графическое окно подсказывает какую последовательность индексов должен ввести пользователь, чтобы переместить фокус на требуемый элемент. Например, на рис. 33 показан 6-мерный массив, поэтому подсказка содержит шесть цифр – «…: 0,1,2,3,4,5». Например, для перехода к элементу array6[20][1][8][2][5][4] надо ввести последовательность «20,1,8,2,5,4».

### 15.3 Статистическая информация по значениям. Вкладка «Statistics»

На данной вкладке отображается информация по минимальному, среднему и максимальному значениям, количеству элементов со значениями «бесконечность», «нечисло», а также элементов меньше, равных или больше нуля и т.д.

### 15.4 Графическое отображение значений

Параллельный отладчик может представить значения 1- и 2-мерного массива в виде плоского или объемного изображения (рис. 35). Если у массива больше 2-х размерностей, то кнопка «Отобразить» («Visualize») будет заблокирована. В процессе формирования изображения параллельный отладчик учитывает диапазоны индексов размерностей.

Графическое окно, пример которого дан на рисунке ниже, позволяет вращать изображение с помощью мыши или с помощью клавиш клавиатуры. Клавишами «-» и «+» можно изменять масштаб графического изображения.

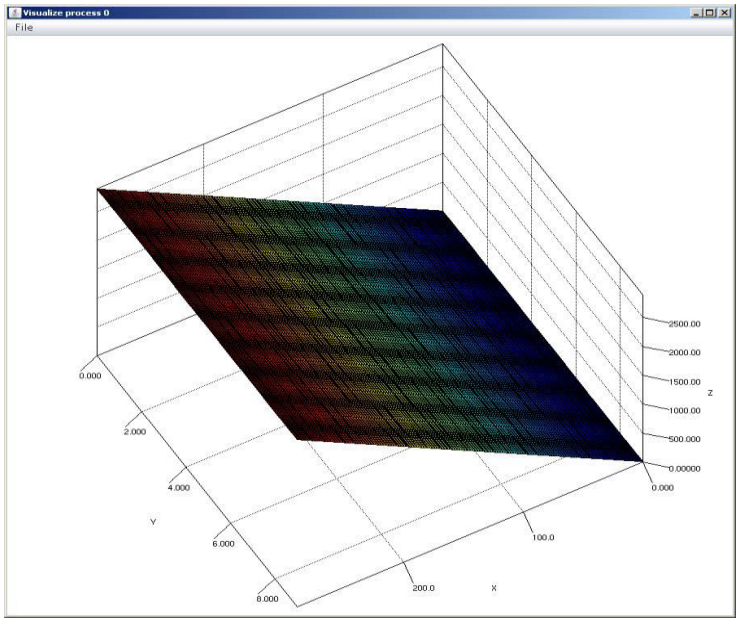



Рисунок 35 – Пример 3-D изображения значений элементов массива

Изображение можно записать в файл, если выбрать пункт «Сохранить» («Save») в меню «Файл» («File»). В появившемся на экране дисплея диалоге пользователь должен выбрать или ввести название файла, в который должно быть записано изображение.

### 15.5 Запись массива в файл

Пользователь может записать значения элементов массива в файл формата CSV или HDF5 для последующего анализа в офисной или специализированной математической программе. При экспорте параллельный отладчик учитывает диапазоны индексов размерностей.

## 16. СРАВНЕНИЕ ПЕРЕМЕННЫХ В ПРОЦЕССАХ И ПОТОКАХ

Сравнить значения программных переменных в разных процессах и потоках позволяют два графических окна – «Межпроцессное сравнение» и «Межпотоковое сравнение» («Cross-process comparision» и «Cross-thread comparision»)[6]. Данные графические окна могут быть вызваны из пункта меню «Инструменты» («Tools»), а также с помощью вызова всплывающего меню на вкладке с содержимым исходного файла и всплывающего меню таблицы на вкладке «Переменные» («Locals»). По наполнению и пользовательскому интерфейсу графические окна примерно одинаковы, поэтому здесь приведено описание графического окна, в котором сравнивается значение программной переменной в разных процессах (рис. 36).

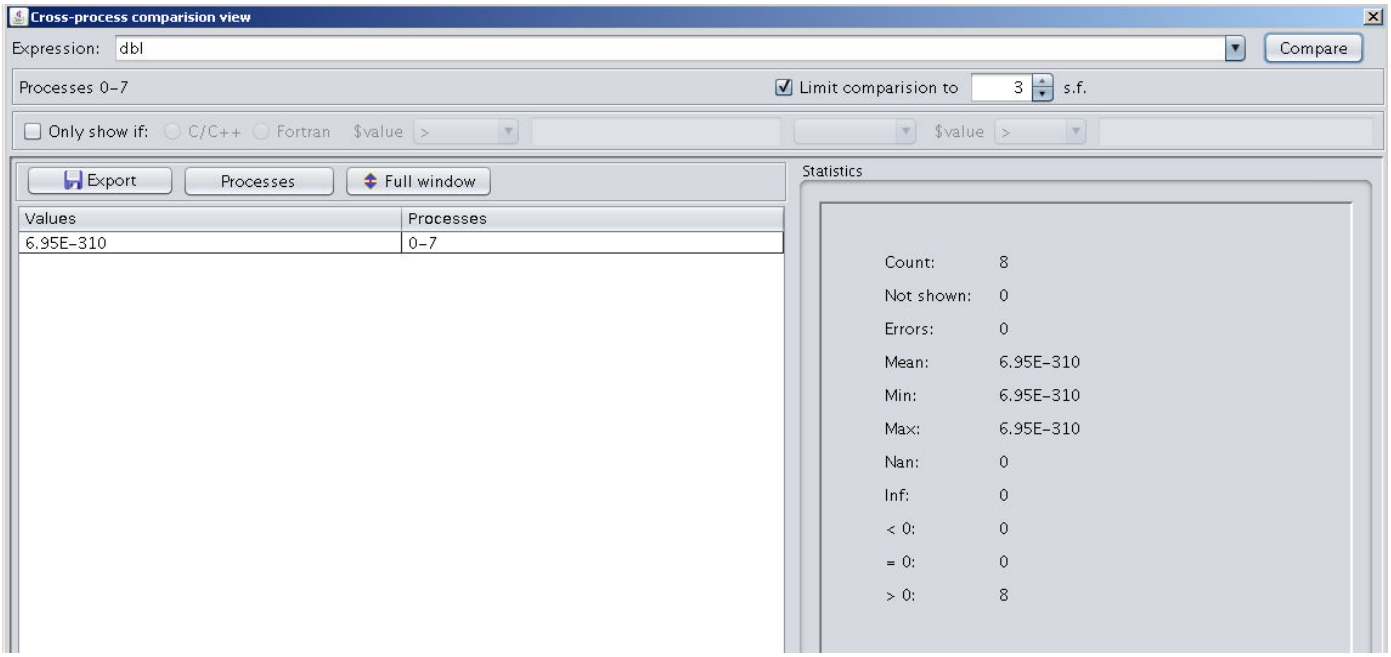



Рисунок 36 – Пример графического окна сравнения значения программной переменной в разных процессах

Диапазон процессов формируются из рангов процессов выбранной группы в режиме отладки «Группа» («Group») или отладки процессов в режиме «Процесс» («Process»). Если в сравнении используется программная переменная с плавающей точкой, то с

[6] Выполняется сравнение значений в программных потоках, исполняющихся на хосте, а не на ускорителе.

помощью графического объекта с названием «Ограничить сравнение» («Limit comparision to») можно ограничить точность сравнения нужным количеством знаков после точки.

На графической панели «Статистика» («Statistics») отображается минимальное, среднее и максимальное значения переменной, а также количество значений типа «бесконечность», «нечисло» и т.д.

### 16.1 Фильтрация значений

Если необходимо, чтобы в таблице отображались цифровые значения, удовлетворяющие определенному критерию (больше или меньше нуля и т.д.), то фильтрацию информации можно осуществить с помощью установки галки в графическом объекте «Показать если только» («Only show if») и инициализации условных выражений. Например, если указано условие $value == 0 («показать все значения, которые равны нулю»), то все отличные от 0 значения будут скрыты.

### 16.2 Запись информации в файл

Записать результаты сравнения в файл формата CSV можно посредством всплывающего меню, которое вызывается щелчком по правой кнопке мыши на таблице с результатами сравнения.

### 16.3 Изменение набора процессов

Чтобы сравнить значения программной переменной в другом наборе рангов процессов, наобходимо щелкнуть мышью по кнопке «Процессы» («Processes») и в появившемся на экране дисплея графическом окне ввести новый числовой диапазон. Запятая является разделителем. Пример: 0-23,28,30-101.

# 17. ПРОФИЛИРОВАНИЕ ПРОГРАММЫ

## 17.1 Настройки профилирования

Для профилирования программы пользователь может использовать профилировщики Google Performance Tools (GPT) [3] и mpiP [4], с помощью которых можно проанализировать производительность программы (CPU-profiler), собрать информацию об использовании программной «кучи» (Heap-profiler) и функций MPI (mpiP). Далее Heap-profiler будет называться профилировщиком динамической памяти, используемой программой.

GPT - это «семплирующие» профилировщики, которые собирают метрики программы с некоторой частотой, например, профилировщик производительности выполняет сбор характеристик работы программы каждые 100 Гц (значение по умолчанию).

Код профилировщиков GPT может быть скомпанован с программой, кроме этого они могут быть использованы как разделяемые программные библиотеки (режим по умолчанию).

Настройка профилировщиков выполняется на вкладке «Профилировщики» («Profilers») при конфигурировании задания или формировании сессии отладки на локальном (инструментальном) компьютере ВС.

Профилировщик производительности можно настроить на автоматический старт после запуска программы (если сигнал не указан), а также его можно включить/выключить по сигналу в нужный момент выполнения программы. На рис. 37 показано, что данный профилировщик может быть включен по сигналу 30. Обратите внимание, что если программа не собрана с профилировщиком, то необходимо установить галку в «Предварительная загрузка библиотек профилирования производительности и динамической памяти» («Preload CPU- and Heap-profiler library»).

Профилировщик динамической памяти программы по достижении указанного в «Интервал распределения» («Allocation interval») или «Интервал использования» («In-

use interval») количества распределенных/занятых мегабайтов оперативной памяти записывает в каталог /tmp результаты профилирования в файлы (равные по умолчанию 1000Мб). Если свободного пространства в /tmp мало, то рекомендуется увеличить значение интервала, чтобы в процессе профилирования было создано меньше файлов. Параллельный отладчик отображает информацию последнего файла с результатами профилирования.

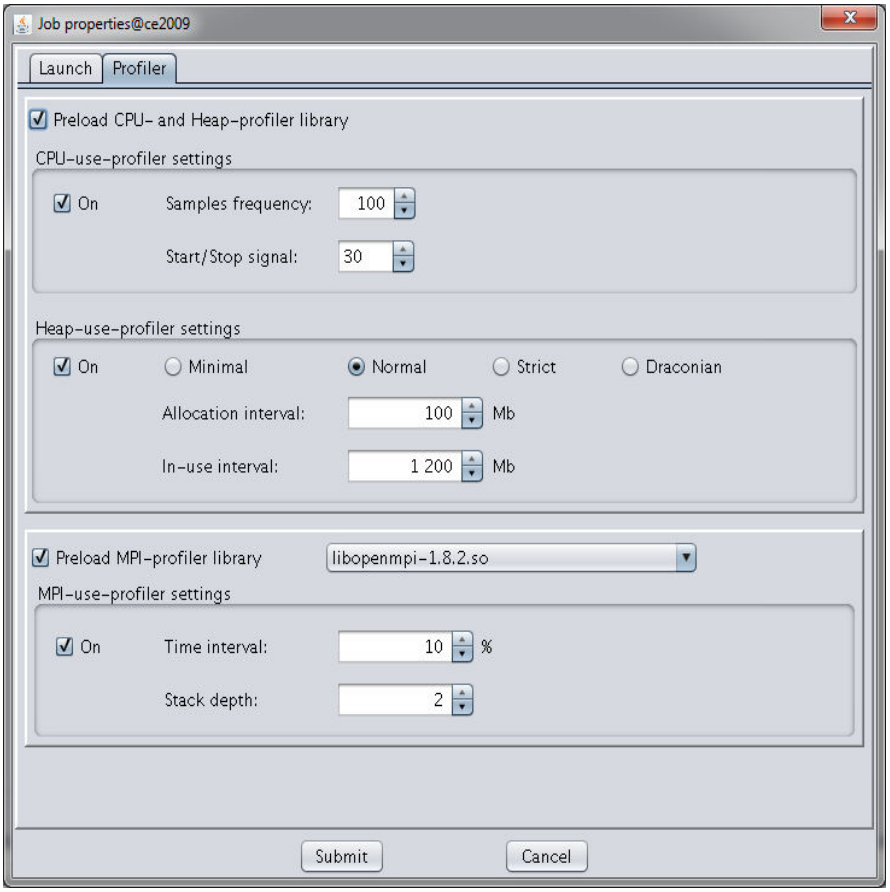



Рисунок 36 – Пример графического окна, в котором настраиваются профилировщики

В случае необходимости проанализировать информацию из предыдущих файлов, воспользуйтесь утилитой pprof, которая входит в состав GPT.

Профилировщик динамической памяти можно использовать в нескольких режимах контроля утечек памяти в программе:

- Minimal – «минимальный», проверка динамической памяти начинается при инициализации программы в функциях/процедурах, которые вызываются перед функцией main(). Если утечки памяти происходят в одноразовых глобальных инициализаторах, то «минимальный» режим полезен, иначе следует использовать другие режимы.

- Normal – «нормальный», режим по умолчанию, в котором контролируются любые динамически создаваемые в памяти объекты. Сообщается об утечке памяти, если в момент завершения программы не доступны любые данные, которые используются в таких программных объектах.

- Strict – «строгий» режим, аналогичен «нормальному», но имеет несколько дополнительных проверок. Например, проверяется возможность потерять выделенную программную память в глобальных деструкторах. Так, если в программе есть глобальная переменная, которая указывает на часть памяти в процессе выполнения программы, но затем «забывает» о ней, не вызывая функцию free(), а в программе её приравнивают к NULL. Сообщение об этом выводится в «строгом» режиме.

- Draconian – осуществляет «безжалостную» проверку, суть которой заключается в том, что перед завершением программы должна быть освобождена вся память, выделенная под её программные структуры/объекты.

Все файлы с результатами профилирования после завершения отладочной сессии из каталога /tmp удаляются.

Для использования профилировщика MPI необходимо установить галку в графическом объекте «Предварительная загрузка библиотеки профилировщика MPI» («Preload MPI-profiler library») и выбрать название разделяемой библиотеки mpiP, которая соответствует используемой в программе реализации MPI. Значения «Временной порог» («Time interval») и «Глубина стека» («Stack depth») можно оставить без изменений – 10% и 2 соответственно.

### 17.2 Управление и просмотр результатов использования памяти и процессора

После завершения программы можно просмотреть результаты профилирования, используя подпункты «Профилировщик производительности» и «Профилировщик динамической памяти» («CPU-profiler» и «Heap-profiler») меню «Инструменты» («Tools»). На рис. 38 показано графическое окно, в котором отображаются результаты профилирования производительности программы.

Столбец «Call» в таблице содержит количество вызовов некоторой строки функции, если указан режим отображения «Функции» («Functions»). Если включен режим «Files» (по файлам), то данный столбец содержит сумму обращений к коду из отдельного файла. Соответственно, если указан режим «Lines» (по строкам), то количество вызовов строки в отдельном файле.

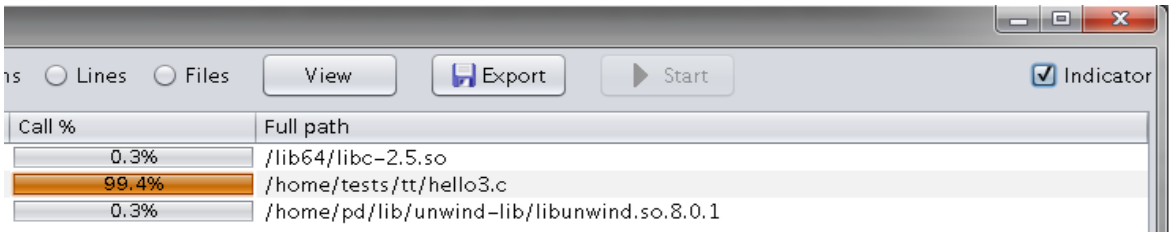



Рисунок 38 – Пример графического окна профилирования производительности программы, показан режим «по функциям»

Столбец «Call %» содержит проценты от общего количества вызовов.

Внимание, если пользователю нужно перейти на указанную в столбце «Позиция» («Position») строку исходного текста программы, то нужно щелкнуть мышью по требуемой строке таблицы.

Серая (отключенная) кнопка запуска профилирования в верху рисунка сообщает, что в настройках профилировщика не указан программный сигнал. Если бы он использовался, то данная кнопка была бы доступна для нажатия.

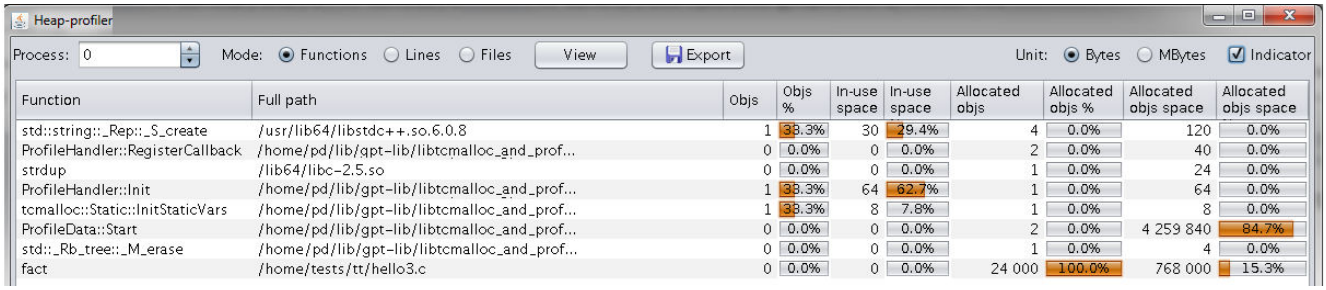
Если для включения/выключения профилировщика производительности (CPU-profiler) был указан программный сигнал, то для запуска этого планировщика в некотором процессе пользователь должен вызвать данное графическое окно, выбрать

ранг процесса, а потом запустить профилировщик, щелкнув мышью по кнопке запуска («Start»).

Для профилирования программных потоков следует перейти в режим отладки потоков, расставить точки останова в нужных местах завершения потока, включить профилирование, а по достижении точки выхода из программного потока его выключить. Есть другое решение – использовать GPT API в программе.

Кнопка «Экспорт» («Export») позволяет записать информацию, которая находится в таблице на рис. 38, в файл формата CSV.

На рис. 39 показан пример результатов профилирования динамической памяти.




| Function | Full path | Objs | Objs % | In-use space | In-use space | Allocated objs | Allocated objs % | Allocated objs space | Allocated objs space |
|---|---|---|---|---|---|---|---|---|---|
| std::string::_Rep::_S_create | /usr/lib64/libstdc++.so.6.0.8 | 1 | 33.3% | 30 | 29.4% | 4 | 0.0% | 120 | 0.0% |
| ProfileHandler::RegisterCallback | /home/pd/lib/gpt-lib/libtcmalloc_and_prof... | 0 | 0.0% | 0 | 0.0% | 2 | 0.0% | 40 | 0.0% |
| strdup | /lib64/libc-2.5.so | 0 | 0.0% | 0 | 0.0% | 1 | 0.0% | 24 | 0.0% |
| ProfileHandler::Init | /home/pd/lib/gpt-lib/libtcmalloc_and_prof... | 1 | 33.3% | 64 | 62.7% | 1 | 0.0% | 64 | 0.0% |
| tcmalloc::Static::InitStaticVars | /home/pd/lib/gpt-lib/libtcmalloc_and_prof... | 1 | 33.3% | 8 | 7.8% | 1 | 0.0% | 8 | 0.0% |
| ProfileData::Start | /home/pd/lib/gpt-lib/libtcmalloc_and_prof... | 0 | 0.0% | 0 | 0.0% | 2 | 0.0% | 4 259 840 | 84.7% |
| std::_Rb_tree::_M_erase | /home/pd/lib/gpt-lib/libtcmalloc_and_prof... | 0 | 0.0% | 0 | 0.0% | 1 | 0.0% | 4 | 0.0% |
| fact | /home/tests/tt/hello3.c | 0 | 0.0% | 0 | 0.0% | 24 000 | 100.0% | 768 000 | 15.3% |

Рисунок 39 – Пример графического окна профилирования памяти

Столбцы таблицы, показанной на рисунке, содержат/означают следующее:

«Function» - название функции  (просмотр в режиме «Functions»).

«File» - путь и название файла (просмотр в режиме «Files»).

«Position» - путь и название файла с номером строки (просмотр в режиме «Lines»).

«Objs» - количество размещенных в памяти программных объектов.

«Objs %» - содержит процент от количества динамически выделяемой памяти программы, которые занимают объекты некоторой функции.

«In-use space» - содержит количество байтов/мегабайтов выделенной, но не освобожденной (используемой) памяти. Единица измерения зависит от положения переключателя «Единица» («Unit»).

«In-use space %» - содержит проценты от памяти, используемой всеми объектами программы.

«Allocated objs» - содержит количество выделенных/созданных в памяти объектов в некоторой функции.

«Allocated objs %» - содержит процент выделенных/созданных в памяти объектов от их общего количества.

«Allocated objs space» - содержит общее количество распределенных байтов/мегабайтов оперативной памяти, выделенной программным объектам.

«Allocated objs space %» - содержит процент распределенных мегабайтов относительно общего количества выделенной памяти. Единица измерения зависит от положения переключателя «Единица измерения» («Unit»).

С помощью клика по выделенной в таблице строке можно перейти на строку исходного текста программы или открыть исходный файл.

Необходимо отметить, что в некоторых MPI программах при одновременном использовании профилировщиков GPT и mpiP происходит зависание программы на функции MPI_Init(). Эта ситуация исправляется отключением одного из профилировщиков.

### 17.3 Просмотр результатов профилирования MPI

Результаты профилирования MPI[7] можно просмотреть после завершения программы, используя подпункт «Профилировщик MPI» («MPI-profiler») меню «Инструменты» («Tools»). На рис. 40 показано графическое окно, в котором отображаются результаты профилирования MPI. После выбора требуемого пункта в раскрывающемся списке пользователь должен щелкнуть мышью по кнопке «Показать» («View»).

Кнопка с надписью «Экспорт» («Export») позволяет записать информацию, которая находится в таблице, в файл формата CSV.

[7] Возможность профилирования MPI в ознакомительной версии параллельного отладчика заблокирована.

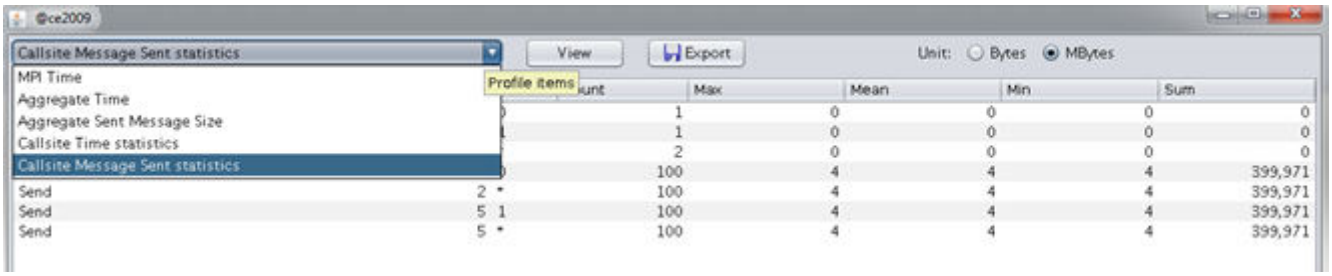



Рисунок 40 – Пример графического окна профилирования MPI

В терминах профилировщика mpiP «callsite» - это вызов MPI функции в программе. Все вызовы имеют собственный цифровой идентификатор.

Таблица «MPI Time» содержит следующие столбцы:

- Task - процесс, ранг процесса параллельной программы.
- Apptime (секунды) - время выполнения в секундах от завершения MPI_Init() до вызова MPI_Finalize().
- MPI_Time - время (в секундах), затраченное на выполнение всех MPI функций, используемых в программе.
- MPI - показывает отношение MPI_Time к Apptime в процентах.

Знак * в таблице свидетельствует о том, что стока содержит сумму значений предыдущих строк.

Таблица «Aggregate Time» содержит информацию о двадцати самых ресурсоемких вызовах. Описание столбцов этой таблицы дано ниже:

- Call - название MPI функции.
- Site - идентификатор вызова, который был ему присвоен mpiP.
- Time - совокупное время в миллисекундах, потраченное на выполнение данной MPI функции в программе.
- App - процентное отношение Time ко времени выполнения программы.
- MPI - отношение Time к полному времени выполнения функций MPI в процентах.

- COV - вариация по времени отдельных процессов для данного Site, коэффициент вариации вычисляется из времени выполнения отдельных процессов.

Таблица «Aggregate Sent Message Size» содержит информацию о двадцати самых ресурсоемких вызовах, описание столбцов таблицы дано ниже:

- Call - название MPI функции.
- Site - идентификатор вызова, который был ему присвоен mpiP.
- Count - количество вызов данной MPI функции в программе.
- Total - всего посланно данных в байтах.
- Avrg - среднее количество данных в байтах, посланных с помощью данной MPI функции.
- MPI - затраты времени на выполнение данной функции к затаратам времени на выполнение всех функций MPI, используемых в программе, в процентах.

Таблица «Callsite Time statistics» содержит следующие столбцы:

- Name - название MPI функции.
- Site - идентификатор вызова, который был ему присвоен mpiP.
- Rank - ранг процесса параллельной программы.
- Count - количество вызовов данной MPI функции в данном программном процессе.
- Max - максимальное время (миллисекунды), затраченное на выполнение данной MPI функции.
- Mean - среднее время (миллисекунды), затраченное на выполнение данной MPI функции.
- Min - минимальное время (миллисекунды), затраченное на выполнение данной MPI функции.
- App - отношение времени выполнения данной MPI функции в программе к её времени выполнения в данном процессе в процентах.
- MPI - отношение времени выполнения вызовов данной MPI функции к полному времени выполнения функций MPI, используемых в данном процессе, в процентах.

Таблицы «Callsite Message Sent statistics» и «Callsite I/O statistics» содержит однотипные столбцы:

- Name - название MPI функции.
- Site - идентификатор вызова, который был ему присвоен mpiP.
- Rank - ранг процесса параллельной программы.
- Count - количество вызовов данной MPI функции в данном процессе.
- Max - максимальный размер сообщения в байтах для данной MPI функции.
- Mean - средний размер сообщения для данной MPI функции в байтах.
- Min - минимальный размер сообщения для данной MPI функции в байтах.
- Sum - суммарное количество байтов, посланных в данном процессе с помощью данной MPI функции.

Если в таблице есть столбец «Site», то двойной щелчок по выделенной строке вызовет переход к строке исходного текста, с которой она связана. Полное описание всех столбцов приведено на домашней странице проекта [4].

### 17.4 Сообщения программы

Сообщение «CPU profile file is not found, check up the checkboxes 'Preload CPU- and Heap-profiler library' and 'CPU-use-profiler setting' in new session window» выдается при нажатии мышью по кнопке «Показать» («View») в графическом окне «Профилировщик эффективности» («CPU profiler»). Программный агент параллельного отладчика не нашел файл с результатами профилирования программы, пользователю предлагается установить галки на закладке «Профилировщики» («Profilers») в графических объектах «Предварительная загрузка библиотек профилирования производительности и динамической памяти» и «Настройки профилировщика производительности» («Preload CPU- and Heap-profiler library» и «CPU-use-profiler setting») в графическом окне создания новой отладочной сессии.

Сообщение «Heap profile file is not found, check up the checkboxes 'Preload CPU- and Heap-profiler library' and 'Heap-use-profiler setting' in new session window» выдается при нажатии мышью по кнопке «Показать» («View») в графическом окне «Профилировщик

динамической памяти» («Heap profiler»). Программный агент параллельного отладчика не нашел файл с результатами профилирования программы, пользователю предлагается установить галки на закладке «Профилировщики» («Profilers») в графических объектах «Предварительная загрузка библиотек профилирования производительности и динамической памяти» и «Настройки профилировщика динамической памяти» («Preload CPU- and Heap-profiler library» и «Heap-use-profiler setting») в графическом окне создания новой отладочной сессии.

Сообщение «Cannot run program "/usr/bin/objdump": error=2, No such file or directory» (нельзя запустить программу … нет такого файла или каталога) появляется на экране дисплея, когда на компьютере, на котором выполняется отладка, в ОС нет утилиты objdump. Необходимо установить данную утилиту, чтобы получить результаты профилирования от программного агента параллельного отладчика.

Сообщение «Cannot run program "/usr/bin/addr2line: error=2, No such file or directory» появляется на экране дисплея, когда на компьютере, на котором выполняется отладка, в ОС нет утилиты addr2line. Необходимо установить данную утилиту, чтобы получить результаты профилирования от программного агента параллельного отладчика.

## 18. ФАЙЛЫ С ШАБЛОНАМИ ЗАДАНИЙ, ЗНАЧЕНИЯМИ АТРИБУТОВ И ИНФОРМАЦИЕЙ ОБ ОТЛАДОЧНОЙ СЕССИИ

Кроме файла с шаблоном задания, который используется отладчиком по умолчанию (системный шаблон), пользователь может создавать задания, используя собственный файл шаблона задания. Для создания задания, например, со спефическими ресурсными требованиями, которое нельзя создать в графическом окне ввода атрибутов задания параллельного отладчика, пользователь может сделать копию системного шаблона задания из подкаталога $PD_HOME/templates, изменить её по своему усмотрению и записать копию файла в каталог $HOME/.pdx. Кроме этого, можно воспользоваться пунктом меню «Шаблон задания» («Template»), с помощью которого создать новый или изменить уже имеющийся шаблон задания в специализированном текстовом редакторе.

При обращении к пункту меню «Кофигурирование и запуск задания» («Configure and start a job») параллельный отладчик предложит пользователю выбрать пользовательский файл шаблона задания, а также загрузит сохраненные ранее значения атрибутов задания из файла с расширением properties, если, конечно, пользователь уже запускал задание, используя выбранный шаблон. Для файла foo.tmpl значения атрибутов задания будут находиться в файле с названием foo.properties.

Если пользователь откажется выбрать какой-либо шаблон из предложенного списка, то параллельный отладчик будет использоватьтсистемный шаблон (шаблон по умолчанию). Внимание. Команды уравления заданием должны быть обязательно указаны в файле шаблона пользовательского задания, однако пользователь может изменять их для своих целей.

По тексту ниже приведена информация, которая может находиться в файле шаблона задания для системы пакетной обработки заданий JAM [5].

```
#
# JAM batch system template file
#
# submitjob=/usr/local/jam/maui/bin/mauisubmit -
# canceljob=/usr/local/jam/maui/bin/canceljob :JOBID
# regexpjob=(\d+\.\w+)
# showqueue=/usr/local/jam/maui/bin/showq
#
Account == debug
JobType == mpi
Output == ./${MAUI_JOB_ID}.out
Error == ./${MAUI_JOB_ID}.err
JobName == debugging
IWD == :IWD
```

Comment == :COMMENT

WCLimit == :WALLTIME

Geom == 1x12:BIGMEM, 5x1, 20x10, 1x2:NVIDEA, 2x12:BIGMEM

Exec == :EXEC  :ARGS

Input == :STDIN

Env == :ENV

Закомментированные строки после надписи «JAM batch system template file» (submitjob, canceljob и т.д.) это - обязательная служебная информация, содержащая шаблоны команд управления и контроля задания. Они содержат путь и название утилит постановки, удаления и команду просмотра очереди. Директива regexpjob служит для определения идентификатора задания. Ключевое слово :JOBID используется для подстановки в занимаемую им позицию идентификатора задания.

Если в системе управления заданиями организована отдельная очередь для интерактивной отладки, то её название можно указать непосредственно в команде submit или в атрибутах задания, например, Account == debug.

Внимание. В директиве Geom указано количество узлов с необходимыми атрибутами (не используется :GEOM): 1*12+5*1+20*10+1*2+2*12=243. Пользователю в графическом окне формирования задания необходимо указать, что заданию требуется 243 процессорных ядра. Например, в графическом объекте ввода атрибутов задания «Количество узлов» можно указать 243, а в «Количестве процессов на узел» 1. Ниже приведены директивы, которые пользователь может применять в файле шаблона задания.

Директивы, указанные в шаблоне задания, заменяются параллельным отладчиком на информацию, которую пользователь вводит в соответствующие графические объекты графического окна с атрибутами задания.

:ARGS - входные параметры, передаваемых исполняемому файлу программы.

:COMMENT - комментарий к заданию.

:ENV - данную директиву параллельный отладчик заменит на указанные пользователем переменные окружения.

:EXEC - директива заменяется на название исполняемого файла отлаживаемой программы.

:IWD - рабочий каталог задания.

:NODES - количество узлов ВС, которое требуется для отладки программы.

:PPN - количество процессоров на узел.

:PPC - количество сопроцессоров на узел.

:STDIN - название файла стандартного ввода.

:WALLTIME - время, которое требуется затратить на отладку программы.

В шаблоне задания для SLURM и Open PBS в случае использования переменных окружения, значения которых связаны между собой (A=0 B=$A C=$B), нужно экпортировать переменные окружения несколько раз, чтобы все переменные были проинициализированы корректно, например:

```
export :ENV

export :ENV
```

Если в течение отладочной сессии пользователь устнавливал точки останова, наблюдения и/или контролировал изменение переменных, то по завершении отладочной сесси параллельный отладчик сохранит информацию о них в файле с расширением session. Таким образом, после завершения отладочной сессии с использованием пользовательского файла шаблона задания foo.tmpl в подкаталоге .pdx домашнего каталога пользователя появятся еще два файла – foo.properties и foo.session.

# 19. ОПЦИИ КОМПИЛЯТОРА. РЕКОМЕНДАЦИИ

Для отладки программы с помощью параллельного отладчика скомпилируйте её с ключом *–g* и с минимальным уровнем оптимизации (*-O0*) или вобще без неё, поскольку в случае использования оптимизации компилятор может переставить инструкции, тогда выполнение программы в отладчике PD будет «скакать» по строкам исходного текста, или вобще не включить отладочную информацию в исполняемый файл программы.

Не используйте ключ *-fomit-frame-pointer*, так как в этом случае компилятор компилирует функции программы так, чтобы они работали без кадра стека, поэтому GDB не сможет отлаживать программу.

Если во вкладке «Locals» нет переменных, то необходимо перекомпилировать программу с ключем *-fno-omit-frame-pointer* (разрешает использование регистра указателя стека процессора).

Оптимизация, которая выполняется компилятором фирмы Intel, когда используется ключ *–ax*, не позволяет получить нужную информацию из стека при отладке программы, так как указатель кадра стека многократного переиспользуется при выполнении программы.

Некоторые компиляторы фирмы Intel не генерируют расширенную отладочную информацию, поэтому используйте ключ *-debug all.*

Компилятор GNU Fortran не включает информацию о модулях в исполняемый файл Фортран программы, поэтому при отладке скомпилированной им программы вкладка «Фортран модули» («Fortran modules») не отображается.

# 20. ОТЛАДКА ПРОГРАММ, ИСПОЛЬЗУЮЩИХ CUDA GPU

Параллельный отладчик позволяет отлаживать программы, использующие NVIDIA CUDA, с возможностью одновременной отладки кода, исполняющегося как в GPU (graphics processing unit), так и на хосте.

## 20.1 Подготовка исполняемого файла программы к отладке

Чтобы отладить программу на GPU, пользователь должен добавить ключи в командную строку компилятора, благодаря которым в исполняемый файл программы добавляется отладочная информация. При использовании компилятора nvcc фирмы NVIDIA ядро должно быть собрано с флагами –g –G. Флаги заставляют компилятор генерировать информацию для отладки ядра и выключают оптимизацию, препятствующую отладке. Для того чтобы использовать в параллельном отладчике отладку памяти с CUDA 5.5 или 6.0, во входных параметрах командной строки nvcc надо указать --cudart shared.

## 20.2 Создание отладочной сессии

Создание отладочной сессии для отладки кода для GPU выполняется также, как описано в пунктах 4.1 и 4.2 данного документа.

Присоединение к уже запущенному процессу, который использует GPU, не возможно, если программа уже запустила ядро или выполнила любую функцию из библиотеки CUDA. Внимание. В программах, использующих MPI, лучше выполнять инициализацию CUDA после вызова MPI_Init.

## 20.3 Отладка программных потоков

Управление программными потоками выполняется в обычной для параллельного отладчика манере: пользователь имеет возможность выполнять, останавливать, ставить контрольные точки и т.д.

При выполнении кода GPU в графическом интерфейсе параллельного отладчика на вкладке «Потоки» («Threads») появляется таблица с перечнем нитей CUDA. Пример её наполнения показан на рисунке ниже.

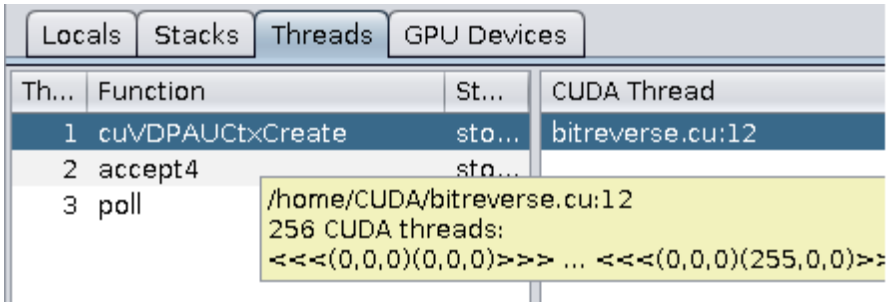



Рисунок 41 – Пример таблицы «CUDA Thread» и всплывающей подсказки с индексами программных нитей

### 20.4 Точки останова

Точки останова действуют на все программные потоки GPU и прерывают выполнение кода всякий раз, когда программный поток достигает точки останова. Однако одновременность срабатывания точек останова в разных потоках зависит от исполняемого кода (нагрузки) на ядро GPU и расписания выполнения, которое составляет внутренний планировщик данного устройства.

Чтобы точки останова действовали в отдельных программных блоках, группах или отдельных потоках, необходимо использовать условие срабатывания, в котором указывать такие встроенные переменные, как индекс потока или блока - threadIdx.x, threadIdx.y, threadIdx.z и т.д.

### 20.5 Выполнение программы по шагам

Отладка программы на GPU заметно отличается по времени от отладки программы, собранной для хоста: выполнение программы для GPU, находящейся под управлением отладчика cuda-GDB не такое быстрое.

Наименьшая единица выполнения на GPU – это «warp», который в NVIDIA GPU составляет 32 программные нити. Выполнить шаг можно только warp(ом).

Параллельный отладчик позволяет отлаживать программные потоки, связанные с отдельными блоками, ядрами или устройствами.

### 20.6 Выполнение программного потока/пауза

Щелчком мышью по кнопке «Запустить/Продолжить» («Run/Continue») в графическом интерфейсе параллельного отладчика можно запустить все программные нити на GPU. Нельзя управлять выполнением отдельного блока, warp(а) или нити.

Щелчок по кнопке «Пауза» остановит выполнение ядра. Операция остановки выполняется не так быстро, как на хосте.

### 20.7 Подсоединение к исполняющемуся приложению

Присоединение к исполняющемуся программному приложению и последующая его отладка возможна для CUDA версии 5 и выше на картах семейства Fermi, включая Tesla C2050/2070, K10, K20 и более современных.

Методика подсоединения к исполняющейся программе была описана в данном документе выше по тексту.

### 20.8 Переключение между ядрами, блоками и нитями

Переключатель нитей (рис. 42) появляется тогда, когда программа достигает кода, выполняемого GPU. Переключатель нитей позволяет пользователю выбрать требуемое ядро, блок и нить. Выбор требуемой нити необходимо завершить посредством щелчка по кнопке «Готово» («Ok»). После этого параллельный отладчик выполнит переключение на выбранную нить.

Первый раскрывающийся список (рисунок ниже) содержит индекс ядра, с помощью следующих трёх формируется индекс блока, а последующие три представляют трехмерный индекс нити внутри блока. Пошаговое выполнение выбранной нити вызывает обновление информации на вкладке локальных переменных, «Оценить выражения» («Evaluate») и перемещение зеленого маркера, отмечающего строку исходного кода программы.

Переключатель нити будет автоматически переключаться на «живую» нить после завершения выбранной ранее нити.


Рисунок 42 – Пример значений переключателя нитей для BlockIdx(0,0,0) ThreadIdx(32,0,0) ядра 0

### 20.9 Информация о GPU

После обнаружения кода для GPU в графическом интерфейсе параллельного отладчика появляется вкладка «Устройства GPU» («GPU Devices»), на которой отображается информация о том, какие процессы используют или сколько устройств

GPU им доступны. На рисунке ниже показано, что процессы с рангами 0-180 могут использовать два GPU-устройства с указанными характеристиками. Процессы с 181 по 800 не используют GPU-устройства (надпись «No device»).

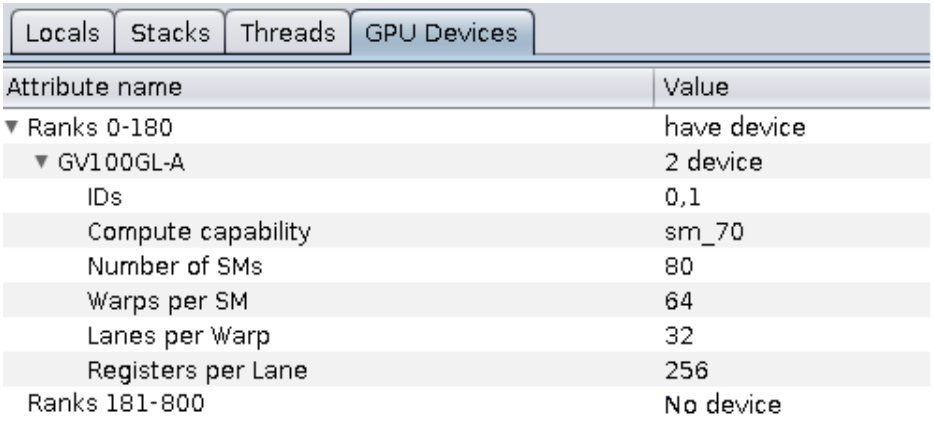



Рисунок 43 – Пример содержимого таблицы «GPU Devices».

# ЗАКЛЮЧЕНИЕ

Графический интерфейс параллельного отладчика почти полностью соответствует графическому интерфейсу зарубежного коммерческого отладчика Allinea DDT. Но, в отличие от Allinea DDT, параллельный отладчик PD – это российский программный продукт, которой невосприимчив к каким-либо зарубежным ограничениям. Кроме этого, он функционирует не только на x86, но и на всех программно-аппаратных платформах, где есть Unix/Linux и среда времени выполнения Java, в частности, на отечественной программно-аппаратной платформе Эльбрус.

## ЛИТЕРАТУРА


1. Allinea DDT and MAP User Guide. Интернет-ресурс. Адрес: http://www.allinea.com.
2. MobaXterm (Unix utilities and X-server on Gnu/Cygwin). Интернет-ресурс. Адрес: http://mobaxterm.mobatek.net.
3. Google Performance Tools. Интернет-ресурс. Адрес: http://code.google.com/p/google-perftools.
4. Профилировщик mpiP. Интернет-ресурс. Адрес: http://mpip.sourceforge.net.
5. Киселёв А. Б., Киселёв С. Н. Система пакетной обработки заданий JAM // Сборник ВАНТ ММФП №4 2009 г. Вопросы атомной науки и техники. Серия: Математическое моделирование физических процессов. Адрес: https://book.sarov.ru/wp-content/uploads/VANT-MMFP-2009-4.pdf