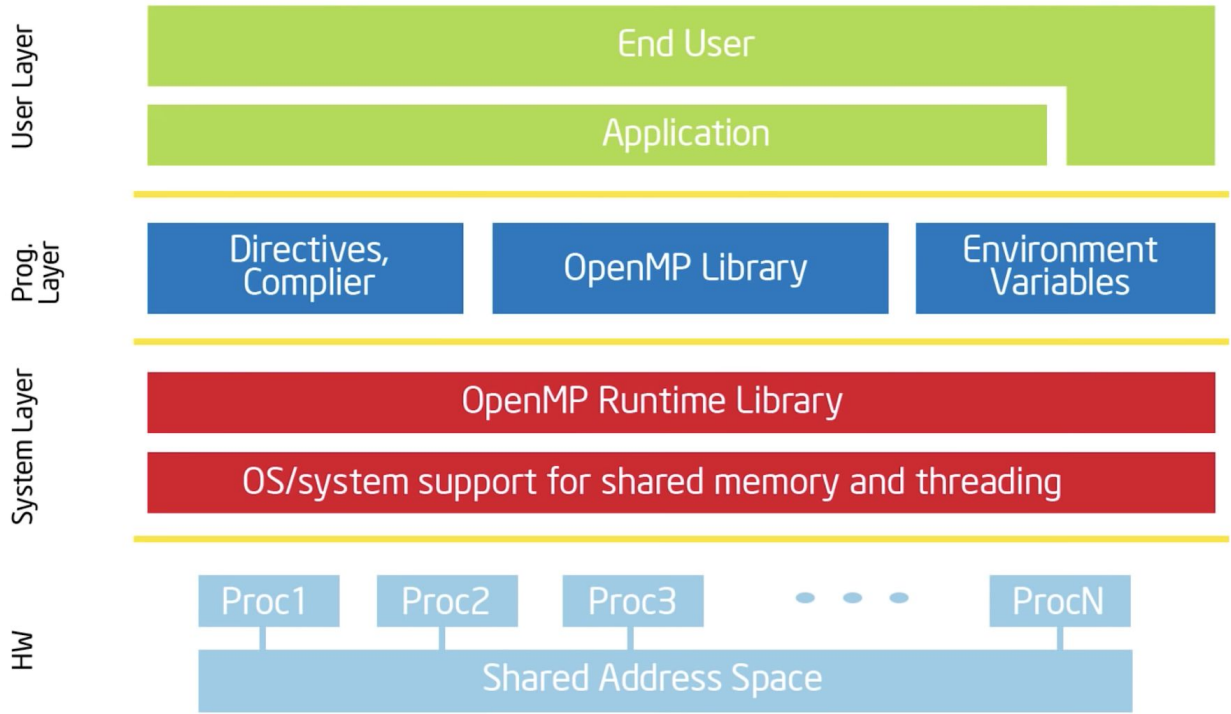
OpenMP

- OpenMP **(Open Multi-Processing)** - открытый стандарт для распараллеливания программ на языках C, C++, Fortran
- Включает в себя набор **директив компилятора**, **библиотечных функций** и **переменных окружения**
- Одно из наиболее популярных средств программирования для **компьютеров с общей памятью**
- OpenMP - **один вариант** программы для **параллельного** и **последовательного** выполнения
- Разработчик стандарта - некоммерческая организация OpenMP ARB, в которую входят представители крупных компаний-разработчиков суперкомпьютеров и программного обеспечения

OpenMP Basic Defs: Solution Stack



Концепция прагм

(из документации Microsoft) *Директивы #pragma предоставляют каждому компилятору способ обеспечения специальных компьютерных функций и функций операционной системы при сохранении общей совместимости с языками C и C++*

pragma - директива компилятора

Если компилятор не распознаёт конкретную прагму, он её игнорирует

Директивы OpenMP оформляются с помощью **#pragma omp ...**

Ключевое слово **omp** используется для того, чтобы исключить случайные совпадения имён директив OpenMP с другими именами в программе

Концепция прагм

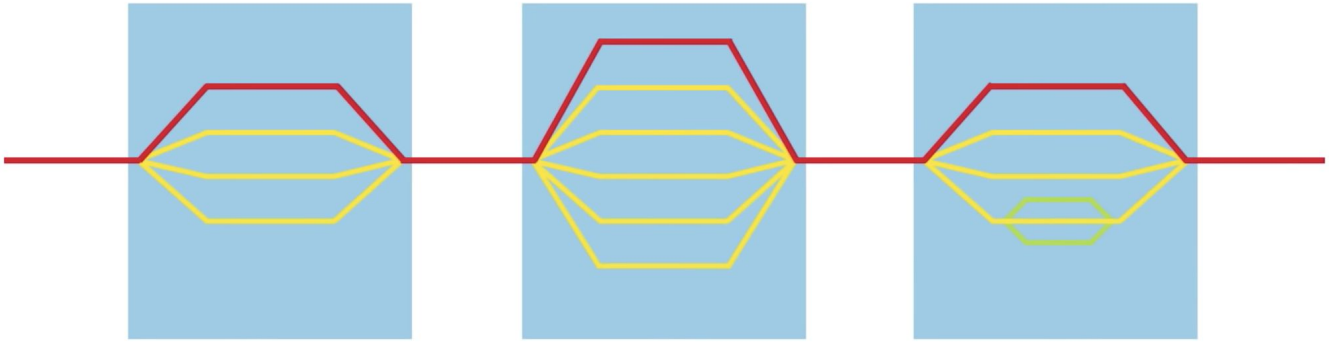
Формат директив на C/C++:

#pragma omp directive-name [опция, ...]

Категории директив OpenMP:

- определение параллельной области
- распределение работы
- синхронизация

Параллельные и последовательные области



Параллельные и последовательные области

В начале работы программы существует **один “основной” поток**

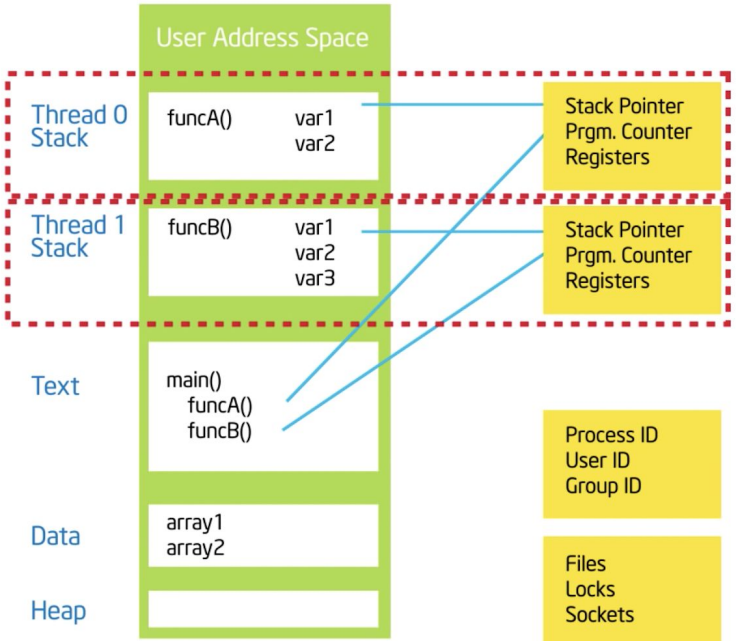
При входе в параллельную часть создаются новые **“рабочие” потоки**, которые уничтожаются при выходе из параллельной части программы

Начало параллельной части программы:
#pragma omp parallel [опция, ...]

В OpenMP переменные в параллельной части программы могут быть:

- shared (общие)
- private (локальные)

Programming Shared Memory Computers



Threads:

* Threads are "light weight processes"
* Threads share Process state among multiple threads. This greatly reduces the cost of switching context.

Hello World на OpenMP

```
omp_get_num_procs();  // получение количества доступных вычислительных ядер

omp_set_num_threads(2);  // явное задание количества потоков

omp_get_num_threads();  // получение количества работающих потоков

omp_get_thread_num();  // получение номера потока
```

Компиляция и запуск

Компиляция:

gcc (g++) -fopenmp filename.c

по умолчанию исполняемый файл имеет имя a.out

Запуск:

./a.out (в общем, как обычно)

В платформах UNIX версия с открытым кодом доступна в проекте компилятора Omni OpenMP (http://www.hpcs.cs.tsukuba.ac.jp/omni-compiler/)

Распараллеливание циклов

Если в параллельной программе встретится цикл, то все его итерации выполнятся всеми потоками

Итерации цикла можно распараллелить между потоками, используя директиву for

#pragma omp for

Эта директива относится к следующему непосредственно за ней оператору for

Синхронизация и критическая секция

Барьер: каждый поток дожидается всех остальных

```
#pragma omp barrier
```

Критическая секция: блок кода, в который единовременно может зайти только один поток

```
#pragma omp critical { критическая секция }
```

Атомарная переменная: безопасное атомарное изменение общей переменной (операции +, *, -, /, &, ^, |, <<, >>)

```
#pragma omp atomic
a = a + 1;
```

Общие правила распараллеливания в OpenMP

1. Найти цикл
2. Модифицировать его так, чтобы итерации не зависели друг от друга
3. #pragma omp parallel for

Ещё полезности:

schedule(type, cnt) - позволяет управлять распределением итераций по циклам (static, dynamic, guided, runtime, auto)

reduction(op:var) - локальная копия var объединяется в результате выполнения цикла в общую копию var с помощью операции op

Распределение работы в циклах

#pragma omp for collapse(n) nowait schedule(type, cnt)

collapse(n) - для n последовательных вложенных циклов общий объём итераций распределяется между потоками
Если опция не задана, директива for относится только к внешнему циклу

nowait - снятие необходимости ожидания всеми потоками друг друга при завершении цикла (по умолчанию в конце цикла есть барьер)

Определение времени работы

double omp_get_wtime(void) - возвращает астрономическое время в секундах, произошедшее с некоторого момента в прошлом
Разность возвращаемых значений покажет время работы участка (Таймеры разных потоков могут не быть синхронизированы и выдавать разные значения)

```
double begin, end, total;
begin = omp_get_wtime();
…
end = omp_get_wtime();
total = end - begin;
```

Задача

Решить определённый интеграл методом трапеций

$$\int_0^1 \frac{4}{1+x^2}\,dx$$

MPI & OpenMP

MPI - работает с параллельными процессами посредством обмена сообщениями, подходит для систем с разделяемой памятью

OpenMP - работает с параллельными потоками, подходит для систем с общей памятью

MPI и OpenMP можно использовать одновременно в одной программе

Полезная литература

Курс видеолекций на youtube:
https://www.youtube.com/watch?v=nE-xN4Bf8XI&list=PLLX-Q6B8xqZ8n8bwjGdzBJ25X2utwnoEG&index=1

Книжка Антонова
“Технологии параллельного программирования MPI и OpenMP”
https://parallel.ru/tech/tech_dev/MPI%26OpenMP

Сайт parallel.ru
https://parallel.ru/tech/tech_dev/openmp.html