Расширение OpenMP 4.0 – условия

```
#pragma omp task depend(dependency-type: list)
```

- Зависимость задачи удовлетворяется, когда ***задача-predecessor*** выполнена:
- Типы зависимостей:
  - ***in****: сгенерированная задача будет зависима от всех* ***ранее сгенерированных*** *задач с типами зависимостей тех же параметров* ***out*** *и* ***inout***
  - ***out / inout:*** *сгенерированная задача будет зависима от всех* ***ранее сгенерированных*** *задач с* ***любыми*** *типами зависимостей тех же параметров (`in`, `out`, `inout`)*

Зависимые параметры могут являться, например, отрезками массивов, e.g. a[10:20]

Расширение OpenMP 4.0 – условия

#pragma omp task depend

```
void process_in_parallel() {
  #pragma omp parallel
    #pragma omp single
    {
      int x = 1;
      for (int i = 0; i < T; ++i) {
        #pragma omp task shared(x) depend(out: x) // T1
          preprocess_some_data(...);
        #pragma omp task shared(x) depend(in: x)   // T2
          do_something_with_data(...);
        #pragma omp task shared(x) depend(in: x)   // T3
          do_something_independent_with_data(...);
     }
```

- T1 должна быть выполнена до начала T2 и T3
- T2 и T3 могут быть выполнены параллельно

Эксплоит?

Расширение OpenMP 4.0 – условия

#pragma omp task depend

```
void process_in_parallel() {
  #pragma omp parallel
    #pragma omp single
    {
      int x = 1;
      for (int i = 0; i < T; ++i) {
        #pragma omp task shared(x) depend(out: x) // T1
          preprocess_some_data(...);
        #pragma omp task shared(x) depend(in: x)   // T2
          do_something_with_data(...);
        #pragma omp task shared(x) depend(in: x)   // T3
          do_something_independent_with_data(...);
     }
```

- T1 должна быть выполнена до начала T2 и T3
- T2 и T3 могут быть выполнены параллельно

Эксплоит? Да, T1 на след. итерации должен будет ждать завершения T2 и T3

Расширение OpenMP 4.0 – условия

```
#pragma omp task priority(priority-value)
```

- Подсказка для среды выполнения о порядке выполнения задач
  - *Неотрицательное целое число (по умолчанию 0)*
  - *Не более, чем max-task-priority ICV*

    *(переменная окружения **OMP_MAX_TASK_PRIORITY**)*
- Между всеми задачами, ожидающими выполнения задачи с бо̲льшим приоритетом являются предпочтительными
- Не рекомендуется делать выводы об очередности выполнения задач исходя из их приоритета!

Расширение OpenMP 4.0 – условия

```
#pragma omp task final(expr)
```

- Ограничение на создание задач в зависимости от условия *expr*
- Особенно актуально для рекурсивных алгоритмов, когда задача на определенной глубине рекурсии становится неэффективной для распараллеливания
- Слияние общих данных может иметь побочные эффекты:

```
void foo(bool arg) {
  int i = 3;
  #pragma omp task final(arg) firstprivate(i)
    i++;
  printf("%d\n", i); // выведет 3 или 4 зависимо от expr
}
```

Расширение OpenMP 4.0 – условия

```
#pragma omp task mergeable
```

- Подсказка для среды выполнения о том, что при необходимости можно выполнить слияние данных таких заданий вместе, если:
  - *Задание не вложенное: условие* ***`if`*** *присутствует и его значение* `false`
  - *Задание вложенное: условие* ***`final`*** *присутствует и его значение true*

- *(Лично мне) пока не удалось найти адекватных примеров того, как условия* `final` *и* `mergeable` *дают какой-либо профит от их использования*

Расширение OpenMP 4.0 – директивы

```
#pragma omp taskloop
```

- Распараллеливание цикла с использованием задач:
  - *Разбить итерации на отрезки*
  - *Создать задание для каждого отрезка*

Расширение OpenMP 4.0 – директивы

```
#pragma omp taskloop
```

- Условия:
  - *shared*, *private*, *firstprivate*, *lastprivate*
  - *default*
  - *collapse*
  - *final*, *untied*, *mergeable*
- **grainsize(size)**
  - *Отрезки имеют размер минимум **size** и максимум **2*size***
- **num_tasks(num)**
  - *Распределить итерации цикла на **num** задач*