OS2 Lab4

checkbox	☑
due date	@September 3, 2021
class	OS
Status	approved

Task

```
Дочерняя нить должна распечатывать текст на экран. Через две секунды после создания дочерней нити, родительская нить должна прервать ее вызовом функции pthread_cancel.
```

Notes

```
#include <stdio.h>
#include <pthread.h>
#include <stdlib.h>
#include <stdbool.h>
#include <unistd.h>

void* printString(void *args){
    while (true){
        printf("botay!\n");
    }
}

int main() {

    pthread_t endlessThread;
    int status = pthread_create(&endlessThread, NULL, printString,
 NULL);

    if (status != 0){
        perror("failed to create new thread");
        exit(-1);
    }

    sleep(2);

    status = pthread_cancel(endlessThread);

    if (status != 0){
        perror("failed to cancel execution of new thread");
        exit(-1);
    }

    return 0;
}
```

Один поток может передать запрос на принудительное завершение другого потока в том же процессе, обратившись к функции `pthread_cancel`.

По умолчанию вызов `pthread_cancel` заставляет указанный поток вести себя так, словно он вызвал функцию pthread_exit с аргументом `PTHREAD_CANCELED.` Однако поток может отвергнуть запрос или как-то иначе отреагировать на него. Обратите внимание, что функция `pthread_cancel` не ждет завершения потока. Она просто посылает запрос.

В момент запуска потока значение его атрибута `cancelability state` устанавливается равным `PTHREAD_CANCEL_ENABLE`. Если поток установит значение этого атрибута равным `PTHREAD_CANCEL_DISABLE`, вызов функции `pthread_cancel` не будет приводить к завершению потока. Вместо этого запрос на принудительное завершение встает в режим ожидания. Когда поток опять разрешит возможность принудительного завершения, он откликнется на ожидающий запрос в ближайшей точке
выхода [Стивенс 12.7]

```
cc -mt [ flag... ] file... -lpthread [ library... ]
       #include <pthread.h>

       int pthread_cancel(pthread_t target_thread);

The pthread_cancel() function requests that target_thread be canceled.

       By default, cancellation is deferred until target_thread reaches a
       cancellation point. See cancellation(5).

       Cancellation cleanup handlers for target_thread are called when the
       cancellation is acted on. Upon return of the last cancellation cleanup
       handler, the thread-specific data destructor functions are called for
       target_thread. target_thread is terminated when the last destructor
       function returns.

       A thread acting on a cancellation request runs with all signals blocked.
       All thread termination functions, including cancellation cleanup handlers
       and thread-specific data destructor functions, are called with all
       signals blocked.

       The cancellation processing in  target_thread runs asynchronously with
       respect to the calling thread returning from pthread_cancel().
```

`Cancel state` (состояние прерывания) определяет, разрешено ли прерывание нити как таковое. Т.е. этот атрибут может иметь два значения – разрешено или запрещено. Если прерывание разрешено, нить немедленно получает сообщение о попытке ее прервать (хотя, в зависимости от `cancel type`, может отреагировать на это сообщение лишь через некоторое время). Если прерывание запрещено, попытки прерывания нити накапливаются. После того, как прерывания все-таки разрешат, нить получит сигналы о накопившихся попытках.

Переключение состояния прерывания осуществляется функцией `pthread_setcancelstate(3C)`.

Первый параметр этой функции входной и может принимать значения `PTHREAD_CANCEL_ENABLE` (прерывание разрешено) и `PTHREAD_CANCEL_DISABLE` (прерывание запрещено). Эти значения – препроцессорные макроопределения, содержащиеся в файле `pthread.h`. Вызов функции с другими значениями первого параметра приведет к ошибке `EINVAL`. Второй параметр функции – выходной, содержит указатель на переменную, в которой будет размещено старое значение типа прерывания. В качестве этого указателя можно передать `NULL`, в этом случае старое значение состояния будет потеряно. По умолчанию, нить создается с разрешенными прерываниями.

Cancel type (тип прерывания) определяет, в какие моменты нить проверяет сообщения о прерываниях. Этот атрибут может принимать два значения – `PTHREAD_CANCEL_DEFERRED` (отложенное прерывание) и `PTHREAD_CANCEL_ASYNCHRONOUS` (асинхронное прерывание). По умолчанию, нить создается с отложенным типом прерываний. Что означает каждое из возможных значений этого атрибута, описывается далее в этом разделе.

Установка типа прерывания осуществляется функцией `pthread_setcanceltype(3C)`. Схема передачи параметров этой функции аналогична `pthread_setcancelstate(3C)`.

Тип и состояние прерывания могут быть заданы в момент создания нити при помощи установки соответствующих полей в структуре `pthread_attr_t`. Обратите внимание, что оба эти атрибута задаются либо в момент создания нити, либо самой нитью. Внешними по отношению к нити средствами их изменить невозможно. Поэтому проверка значений этих атрибутов в определенные моменты времени не может приводить к ошибкам соревнования.

Асинхронное прерывание означает, что библиотека прерывает нить как

The following cancellation points are defined by the system (system-defined cancellation points): creat(2), aio_suspend(3C), close(2), creat(2), getmsg(2), getpmsg(2), lockf(3C), mq_receive(3C), mq_send(3C), msgrcv(2), msgsnd(2), msync(3C), nanosleep(3C), open(2), pause(2), poll(2), pread(2), pthread_cond_timedwait(3C), pthread_cond_wait(3C), pthread_join(3C), pthread_testcancel(3C), putmsg(2), putpmsg(2), pwrite(2), read(2), readv(2), select(3C), sem_wait(3C), sigpause(3C), sigwaitinfo(3C), sigsuspend(2), sigtimedwait(3C), sigwait(2), sleep(3C), sync(2), system(3C), tcdrain(3C), usleep(3C), wait(3C), waitid(2), wait3(3C), waitpid(3C), write(2), writev(2), and fcntl(2), when specifying **F_SETLKW** as the command.

можно скорее (хотя во многих ситуациях не удается гарантировать, чтобы это происходило точно в тот момент, когда другая нить вызвала `pthread_cancel(3C)`). Асинхронное прерывание требует тщательного анализа всех возможных моментов, когда оно может произойти, и обработки всех ситуаций, связанных с прерываниями в неудачные моменты. Так, если прерывание произойдет во время работы с библиотекой, которая не считается *`thread-safe`*, внутренние данные этой библиотеки могут остаться в несогласованном состоянии.

Для корректной работы большинства библиотек в таких условиях простой поддержки многопоточности не достаточно. В страницах системного руководства Solaris уровень поддержки многопоточности описывается несколькими типами атрибутов, которые рассматриваются в лекции 5; безопасность библиотеки или функции при использовании асинронных прерываний описывается атрибутом MT-Level. У библиотеки, которая безопасна для применения в таких условиях, этот атрибут имеет значение Asynchronous-Cancel-Safe. Для всех библиотек и функций, для которых этот атрибут явным образом не указан на соответствующей странице системного руководства, следует предполагать, что они небезопасны для использования в режиме асинхронных прерываний.

Отложенное прерывание означает, что нить получает сообщение о прерывании лишь в определенные моменты, известные как точки прерывания (cancellation point). Эти точки, в свою очередь, делятся на две категории – явные и неявные. Явные точки прерывания – это вызовы функции `pthread_testcancel(3C)`. Неявные точки

```
+-----------------------------+--------------------+----------------------+
|Cancellation Type/State Table |                    |                      |
|Type                         | State              |                      |
|                             | Enabled (Default)  | Disabled             |
+-----------------------------+--------------------+----------------------+
|Deferred (Default)           | Cancellation occurs | All cancellation     |
|                             | when the target    | requests to the      |
|                             | thread reaches a   | target thread are    |
|                             | cancellation point | held pending.        |
|                             | and a cancel is    |                      |
|                             | pending. (Default)
|                             |
|Asynchronous                 | Receipt of a       | All cancellation     |
|                             | pthread_cancel()   | requests to the      |
|                             | call causes        | target thread are    |
|                             | immediate          | held pending; as     |
|                             | cancellation.      | soon as cancellation |
|                             |                    | is re-enabled,       |
|                             |                    | pending              |
|                             |                    | cancellations are    |
|                             |                    | executed             |
|                             |                    | immediately.         |
+-----------------------------+--------------------+----------------------+
```

Reading list

- [ ]