Has anyone successfully installed CUDA 7.5 on Ubuntu 14.04.3 LTS x86_64?

My workstation has two GPUs (Quadro K5200 and Quadro K2200) with the latest NVIDIA Driver installed (Version: 352.41). After downloaded the file `cuda-repo-ubuntu1404-7-5-local_7.5-18_amd64.deb` from CUDA 7.5 Downloads, I try to install it, but it turns out the result as below:

```
root@P700-Bruce:/home/bruce/Downloads# sudo apt-get install cuda
Reading package lists... Done
Building dependency tree
Reading state information... Done
Some packages could not be installed. This may mean that you have
requested an impossible situation or if you are using the unstable
distribution that some required packages have not yet been created
or been moved out of Incoming.
The following information may help to resolve the situation:

The following packages have unmet dependencies:
 cuda : Depends: cuda-7-5 (= 7.5-18) but it is not going to be installed
 unity-control-center : Depends: libcheese-gtk23 (>= 3.4.0) but it is not going to
be installed
                        Depends: libcheese7 (>= 3.0.1) but it is not going to be
installed
E: Error, pkgProblemResolver::Resolve generated breaks, this may be caused by held
packages.
```

I have tried the solution:

1. `sudo apt-get remove nvidia-cuda-*` # remove old nvidia-cuda packages
2. Install unmet dependencies:

```
root@P700-Bruce:/home/bruce/Downloads# apt-get install cuda-7-5
Reading package lists... Done
Building dependency tree
Reading state information... Done
Some packages could not be installed. This may mean that you have
requested an impossible situation or if you are using the unstable
distribution that some required packages have not yet been created
or been moved out of Incoming.
The following information may help to resolve the situation:

The following packages have unmet dependencies:
 cuda-7-5 : Depends: cuda-toolkit-7-5 (= 7.5-18) but it is not going to be
installed
            Depends: cuda-runtime-7-5 (= 7.5-18) but it is not going to be
installed
 unity-control-center : Depends: libcheese-gtk23 (>= 3.4.0) but it is not going to
be installed
                        Depends: libcheese7 (>= 3.0.1) but it is not going to be
installed
E: Error, pkgProblemResolver::Resolve generated breaks, this may be caused by held
packages.

root@P700-Bruce:/home/bruce/Downloads# apt-get install cuda-toolkit-7-5
Reading package lists... Done
Building dependency tree
Reading state information... Done
Some packages could not be installed. This may mean that you have
requested an impossible situation or if you are using the unstable
distribution that some required packages have not yet been created
or been moved out of Incoming.
The following information may help to resolve the situation:

The following packages have unmet dependencies:
 cuda-toolkit-7-5 : Depends: cuda-core-7-5 (= 7.5-18) but it is not going to be
installed
                    Depends: cuda-command-line-tools-7-5 (= 7.5-18) but it is not
going to be installed
                    Depends: cuda-samples-7-5 (= 7.5-18) but it is not going to be
installed
                    Depends: cuda-documentation-7-5 (= 7.5-18) but it is not going
to be installed
                    Depends: cuda-visual-tools-7-5 (= 7.5-18) but it is not going
to be installed
 unity-control-center : Depends: libcheese-gtk23 (>= 3.4.0) but it is not going to
be installed
                        Depends: libcheese7 (>= 3.0.1) but it is not going to be
installed
E: Error, pkgProblemResolver::Resolve generated breaks, this may be caused by held
packages.
```

3. Install and use **aptitude**

My Ubuntu14.04 OS is just installed and have made the software updates and installed the latest Nvidia driver.

Can you give some help? Thanks in advance!

14.04 nvidia cuda

edited Nov 10 at 20:19
wjandrea
2,242 8 25

asked Sep 9 '15 at 9:58
Bruce Yo
76 1 1 7

10 Answers

The installation of CUDA is little a bit tricky. I've followed the following steps and it works for me. You can refer to this link also.

Confirmation of the environment –

1. `lspci | grep -i nvidia` (Confirm that the information of NVIDIA's board is displayed)
2. `uname -m` (make sure that it is a x86_64)
3. `gcc --version` (make sure it is installed)

Installation of CUDA –

1. Download `cuda_7.5.18_linux.run` file from https://developer.nvidia.com/cuda-downloads
2. Run the following command –
   a. `sudo apt-get install build-essential`
   b. `sudo vi /etc/modprobe.d/blacklist-nouveau.conf`
   c. Then, add the following line in that file: `blacklist nouveau option nouveau modeset=0`
   d. `sudo update-initramfs -u`
3. Reboot computer
4. At login screen, press Ctrl + Alt + F1 and login to your user.
5. Go to the directory where you have the CUDA driver, and run
   a. `chmod a+x .`
   b. `sudo service lightdm stop`
   c. `sudo bash cuda-7.5.18_linux.run --no-opengl-libs`
6. During the install –
   a. Accept EULA conditions
   b. Say YES to installing the NVIDIA driver
   c. Say YES to installing CUDA Toolkit + Driver
   d. Say YES to installing CUDA Samples
   e. Say NO rebuilding any Xserver configurations with Nvidia
7. Check if `/dev/nvidia*` files exist. If they don't, do the following –
   a. `sudo modprobe nvidia`
8. Set Environment path variables –
   a. `export PATH=/usr/local/cuda-7.5/bin:$PATH`
   b. `export LD_LIBRARY_PATH=/usr/local/cuda-7.5/lib64:$LD_LIBRARY_PATH`
9. Verify the driver version –
   a. `cat /proc/driver/nvidia/version`
10. Check CUDA driver version
    a. `nvcc –V`
11. Switch the lightdm back on again
    a. `sudo service lightdm start`
12. Ctrl + Alt + F7 and login to the system through GUI
13. Create CUDA Samples –
    a. Go to `NVIDIA_CUDA-7.5_Samples` folder through terminal
    b. `make`
    c. `cd bin/x86_64/linux/release/`

d. `./deviceQuery`

e. `./bandwidthTest`

f. Both tests should ultimately output a 'PASS' in terminal

14. Reboot the system

edited Feb 19 at 8:53
clearkimura
1,987 12 27

answered Feb 19 at 4:38
Avik
51 1 1

Thanks a lot! This finally works on my Asus UX32VD (Optimus laptop with GeForce 620M). I tried everything and everything. Yesterday I could get nvidia-352 working with Bumblebee, but after installing CUDA toolkit, I couldn't run any of the samples (as if I didn't have a CUDA card, and yes, I was using `optirun`). Other drivers spawned me into login loops or black `unity-greeter`! I cannot thank you enough :) – M2X Jul 18 at 1:36

The only thing that I needed to change here was from `option` to `options` within the blacklist nouveau section. – TheM00s3 Aug 10 at 23:42

I have an HP desktop with NVIDIA GeForce GTX 680. Your instruction mostly worked, except that the graphic card driver that comes with the run file (`cuda_7.5.18_linux.run`) causes `lightdm` to quit working after rebooting (after grub, you would see a black screen with endless flashing cursor). My solution was to first uninstall that driver by `sudo apt-get purge nvidia-*` , and install it using the latest run file downloaded from the NVIDIA official website. And it works perfectly. An alternative solution would be something like the solution (A) in askubuntu.com/a/676772/194156 – Xin Aug 29 at 8:19

There are two ways to install suiting CUDA-driver (for Optimus and else built-in graphics-chipsets on hybrid mainboards) - the first described here is the easiest and the second description is more cumbersome but effective too :

A)

```
sudo add-apt-repository ppa:graphics-drivers/ppa
sudo apt-get update
sudo apt-get install nvidia-355 nvidia-prime
sudo reboot
```

B)

Description of method B is here, but already older (explained by user dschinn1001) - this method B is more humblesome and can be risky, but not harmful. :

How can I Install Nvidia Driver GT 520 and Cuda 5.0 in Ubuntu13.04?

The beta-driver-package for Nvidia to download for Linux is here :

http://www.nvidia.de/object/cuda_1_1_beta.html

Method A is more simple, but not clear, how it interacts with xscreensaver and method B is older, but the driver-package is updated too in recent time, and after method B is done, it should work better with xscreensaver conditioned that xscreensaver is installed. (I tested method B on 13.10 and this was working very good, even with xscreensaver. And I think the rest of this thread is up to the hardware.)

In addition and in reference to bumblebee with Optimus-graphics-chipsets these adjustments for bumblebee are necessary too :

How to set up nVidia Optimus/Bumblebee in 14.04

edited Nov 10 at 20:17
wjandrea
2,242 8 25

answered Sep 21 '15 at 19:19
dschinn1001
1,272 1 12 27

Sounds like lp bug 1428972.

User fennytansy added a workaround in comment #10:

```
sudo apt-get install libglew-dev libcheese7 libcheese-gtk23 libclutter-gst-2.0-0
libcogl15 libclutter-gtk-1.0-0 libclutter-1.0-0
```

answered Sep 9 '15 at 23:37
user3813819
11 1

after I run the command screen became black. i can access only tty1 ? Do u know any other solutions? – Karesh Arunakirinathan Oct 12 '15 at 11:55

I successfully installed CUDA using the runfile method. It's a little trickier to setup because your primary graphics driver *also* has to be installed using the runfile method (See Here).

Try installing *just* the driver. This can be done by using the runfile method. It will prompt you

for each portion of the install and you can disable the `GL` libraries and toolkits. The unity control center has been giving me issues as well with due to the CUDA sample's need to use `libGLU.so` instead of `libGL.so`. This is an easy fix when building your own learning examples.

answered Sep 15 '15 at 18:35
asdf
43 9

Try uninstalling the nvidia driver, and directly installing cuda without it. On a fresh Ubuntu 14.04, I followed the instructions from the nvidia website. Aside from verifying compatible versions of things (gcc, kernel), the instructions were:

```
sudo dpkg -i cuda-repo-ubuntu1404_7.5-18_amd64.deb
sudo apt-get update
sudo apt-get install cuda
```

Happily, the correct nvidia driver was installed as a by-product of the steps above.

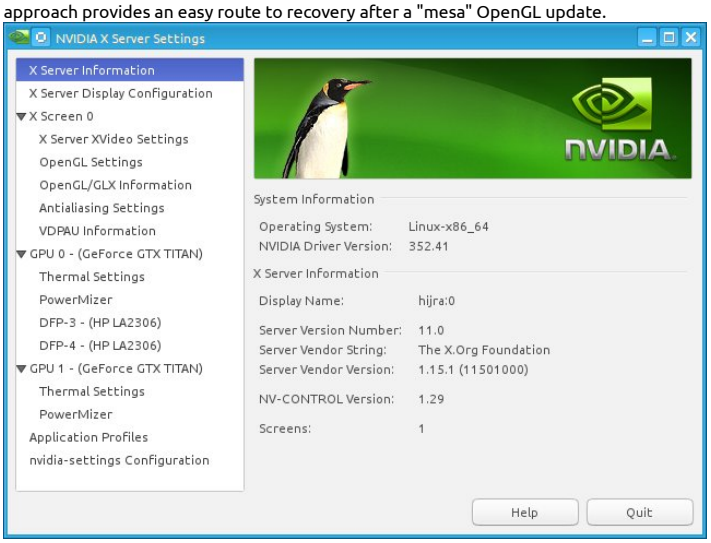
answered Sep 24 '15 at 23:46
661266
11 1

I spent a full day seeking to use "ppa:graphics-drivers/ppa" to update the NVIDIA drivers to version 352. Everything failed. After one install, the gpu-manager.log reported that the drivers were installed while Xorg.0.log would report the opposite.

The nouveau driver had been removed and blacklisted: sudo apt-get --purge remove xserver-xorg-video-nouveau cat /etc/modprobe.d/nouveau-nomodeset-jsrobin.conf blacklist nouveau options nouveau modeset=0 alias nouveau off alias lbm-nouveau off

I finally gave up and used a purely "NVIDIA...bin" solution.

1. Blacklisted nouveau, as shown above.
2. completely uninstalled the nouveau Xserver as cited above.
3. Set the system bios to have PCIe (the two nvidia cards) as primary and deactivated the mainboard HD4600 interface.
4. booted into recovery mode, activated network, then went to console mode.
5. Ran "NVIDIA-Linux-x86_64-352.41.run -uninstall" just to make sure nothing was left.
6. Deleted any old directories in /etc, /usr/local, that looked like a remnant of past cuda or nvidia installs.
7. Ran "NVIDIA-Linux-x86_64-352.41.run"
8. Ran "NVIDIA-Linux-x86_64-352.41.run --check" to verify that everything was correct (it was).
9. Then ran "cuda_7.5.18_linux.run" to complete the install. Things are currently working. Both monitors are up and working. Currently working on building the cuda sample files. Be certain to use the "--help" flags on the NVIDIA install bins. The main reason I decided to go the bin route (along with one of the alternatives not working, is that the "bin" approach provides an easy route to recovery after a "mesa" OpenGL update.



edited Oct 12 '15 at 17:58

answered Sep 18 '15 at 17:05
user225932
111 3

I rebooted Ubuntu today, and found there is another unmet dependency something like `libcog15 : Depends: mesa-driver...` (I cannot remember the full package name), so I used `apt-get install` to install the "mesa-driver". After that, CUDA 7.5 installed successfully.

Note that my Kernel version is **3.19.0-28-generic** and the gcc version is **Ubuntu 4.8.4-2ubuntu1~14.04**, which is not found at CUDA 7.5 official documents. I will check if it really work.

edited Nov 10 at 20:24
wjandrea
2,242 8 25

answered Sep 10 '15 at 1:50
Bruce Yo
76 1 1 7

1 For some reason the mesa driver on my computer caused all sorts of unity issues on boot up and caused a full failure of my system. Be careful. – asdf Sep 18 '15 at 20:34

@Bruce Yo - This is in general not only a matter of mesa, this depends on the chipsets on hybrid nvidia-graphics-cards, which are all different. You should consider my solution too. :o) – dschinn1001 Sep 22 '15 at 17:19

Please refer to: https://github.com/astorfi/Caffe_Deep_Learning/blob/master/Installation/readme.md. It is related to installation of Caffe in essence but it address the CUDA installation as well.

answered Oct 27 at 6:53
amirsina torfi
1

1 Hi @amirsani_torfi, welcome to ask.ubuntu. Note that whilst your link might possibly provide the information needed to address the question asked, links can get removed at anytime. I would suggest that you instead edit your question to include the important information from that link. – Tshilidzi Oct 27 at 7:12

I tried sudo su and apt-get install cuda instead of sudo apt-get install cuda. It worked.

```
sudo dpkg -i cuda-repo-ubuntu1404_7.5-18_amd64.deb
sudo apt-get update
sudo su
apt-get install cuda
```

answered Feb 5 at 2:54
softgearko
1 1

Welcome to Ask Ubuntu, nice to see you sharing knowledge. However, this is not a forum, this is a Q&A site, check this help tour. Duplicating others answer (of 661266 user) does not help, you gonna able to up vote when you get enough reputation. – user.dz Feb 5 at 9:54

@Sneetsher Thank for your comment. I have tried to 661266 user's answer but It didn't work. When I used "su" instead of "sudo", it worked. I don't know why exactly. However, It worked with my trial. I believe that it is worth for somebody while to try my solution. – softgearko Feb 11 at 7:59

-problems with lightdm log in (login loop)

-problems with driver istall ("Driver Installation failed: it appears, that a X server is running...")

To successfully install a NVidia CUDA Toolkit on Ubuntu 16.04 64bit I've just had to do:

1. make a liveImage of Ubuntu on pendrive (8GB pen is enough) - such a try will **save a ton of nerves**, before unsuccessful install on your host Linux system!!!
2. login on live session on pendrive ("Try Ubuntu, before install")
3. add sudo user at live session:

   sudo adduser admin (#pass: admin1)

   sudo usermod -aG sudo admin
4. logout from live session, log in as #admin
5. download CUDA Toolkit from NVidia official site (~1.5GB)
6. change privileges for downloaded installer file (DO NOT INSTALL AT THIS STEP!):
   sudo chmod +x cuda_X.X.run
7. switch to console view:

   Ctr+Alt+F1 (to switch on terminal view) Ctr+Alt+F7 (to switch from terminal view to graphical server)
8. at console view (Ctr+Alt+F1) log in:

   login: admin pass: admin1
9. stop graphical running service:

sudo service lightdm stop

10. check if graphical server is off - after switching Ctr+Alt+F7 the monitor should be blank black, switch back on console view Ctr+Alt+F1
11. install CUDA Toolkit, with such configuration:

    sudo ./cuda_X.X.run (press 'q' for license read skip) do not install OpenGL library do not update system X configuration other options make yes and paths as default
12. turn on graphical server:

    sudo service lightdm start
13. log in as user (if you automatically log in as #ubuntu at live session log out):

    login: admin pass: admin1
14. check whatever nvcc compiler works with provided simple parallel vector sum at GPU Blocks:

    save vecSum.cu and book.h at new files, compile and run at terminal: /usr/local/cuda-8.0/bin/nvcc vecSum.cu && clear && ./a.out
15. check console printout - it should be similar to: 0.000000 + 0.000000 = 0.000000

```
-1.100000 + 0.630000 = -0.000000

-2.200000 + 2.520000 = 0.319985

-3.300000 + 5.670000 = 2.119756
-4.400000 + 10.080000 = 5.679756
-5.500000 + 15.750000 = 10.250000
-6.600000 + 22.680000 = 16.017500
-7.700000 + 30.870001 = 23.170002
-8.800000 + 40.320000 = 31.519997
-9.900000 + 51.029999 = 41.129967
```

16. **if everything went well on pendrive live session, do the same on your host linux system**

P.S. Please note that it is not ideal tutorial, but works fine for me!

======= vecSum.cu =====

```
#include "book.h"
#define N 50000
///usr/local/cuda-8.0/bin/nvcc vecSum.cu && clear && ./a.out

//"HOST" = CPU
//"Device" = GPU

__global__ void add( float *a, float *b, float *c )
{
    int tid = blockIdx.x;
    if ( tid < N )
        c[ tid ] = a[ tid ] + b[ tid ];
}

int main ( void )
{
    float a[ N ], b[ N ], c[ N ];
    float *dev_a, *dev_b, *dev_c;
    //GPU memory allocation
    HANDLE_ERROR( cudaMalloc( ( void** )&dev_a, N * sizeof( float ) ) );
    HANDLE_ERROR( cudaMalloc( ( void** )&dev_b, N * sizeof( float ) ) );
    HANDLE_ERROR( cudaMalloc( ( void** )&dev_c, N * sizeof( float ) ) );

    //sample input vectors CPU generation
    for ( int i = 0; i < N; i++ )
    {
        a[ i ] = -i * 1.1;
        b[ i ] = i * i * 0.63;
    }

    //copy/load from CPU to GPU data vectors a[], b[] HostToDevice
    HANDLE_ERROR( cudaMemcpy( dev_a, a, N * sizeof( float ),
cudaMemcpyHostToDevice ) );
    HANDLE_ERROR( cudaMemcpy( dev_b, b, N * sizeof( float ),
cudaMemcpyHostToDevice ) );

    //calculate sum of vectors on GPU
    add<<<N,1>>> ( dev_a, dev_b, dev_c );

    //copy/load result vector from GPU to CPU c[] DeviceToHost
    HANDLE_ERROR( cudaMemcpy( c, dev_c, N * sizeof( float ),
cudaMemcpyDeviceToHost ) );

    //printout results
    for ( int i = 0; i < 10; i++ ) printf( "%f + %f = %f\n", a[ i ], b[ i ], c[ i
] );

    //free memory and constructed objects on GPU
    cudaFree( dev_a );
    cudaFree( dev_b );
    cudaFree( dev_c );

    return 0;
}
```

========= book.h ======

```
/*
 * Copyright 1993-2010 NVIDIA Corporation.  All rights reserved.
 *
 * NVIDIA Corporation and its licensors retain all intellectual property and
 * proprietary rights in and to this software and related documentation.
 * Any use, reproduction, disclosure, or distribution of this software
 * and related documentation without an express license agreement from
 * NVIDIA Corporation is strictly prohibited.
 *
 * Please refer to the applicable NVIDIA end user license agreement (EULA)
 * associated with this source code for terms and conditions that govern
 * your use of this NVIDIA software.
 *
 */


#ifndef __BOOK_H__
#define __BOOK_H__
#include <stdio.h>

static void HandleError( cudaError_t err,
                         const char *file,
                         int line ) {
    if (err != cudaSuccess) {
        printf( "%s in %s at line %d\n", cudaGetErrorString( err ),
                file, line );
        exit( EXIT_FAILURE );
    }
}
#define HANDLE_ERROR( err ) (HandleError( err, __FILE__, __LINE__ ))


#define HANDLE_NULL( a ) {if (a == NULL) { \
                            printf( "Host memory failed in %s at line %d\n", \
                                    __FILE__, __LINE__ ); \
                            exit( EXIT_FAILURE );}}

template< typename T >
void swap( T& a, T& b ) {
    T t = a;
    a = b;
    b = t;
}


void* big_random_block( int size ) {
    unsigned char *data = (unsigned char*)malloc( size );
    HANDLE_NULL( data );
    for (int i=0; i<size; i++)
        data[i] = rand();

    return data;
}

int* big_random_block_int( int size ) {
    int *data = (int*)malloc( size * sizeof(int) );
    HANDLE_NULL( data );
    for (int i=0; i<size; i++)
        data[i] = rand();

    return data;
}


// a place for common kernels - starts here

__device__ unsigned char value( float n1, float n2, int hue ) {
    if (hue > 360)      hue -= 360;
    else if (hue < 0)   hue += 360;

    if (hue < 60)
        return (unsigned char)(255 * (n1 + (n2-n1)*hue/60));
    if (hue < 180)
        return (unsigned char)(255 * n2);
    if (hue < 240)
        return (unsigned char)(255 * (n1 + (n2-n1)*(240-hue)/60));
    return (unsigned char)(255 * n1);
}

__global__ void float_to_color( unsigned char *optr,
                                const float *outSrc ) {
    // map from threadIdx/BlockIdx to pixel position
    int x = threadIdx.x + blockIdx.x * blockDim.x;
    int y = threadIdx.y + blockIdx.y * blockDim.y;
    int offset = x + y * blockDim.x * gridDim.x;

    float l = outSrc[offset];
    float s = 1;
    int h = (180 + (int)(360.0f * outSrc[offset])) % 360;
    float m1, m2;

    if (l <= 0.5f)
        m2 = l * (1 + s);
    else
        m2 = l + s - l * s;
    m1 = 2 * l - m2;

    optr[offset*4 + 0] = value( m1, m2, h+120 );
    optr[offset*4 + 1] = value( m1, m2, h );
    optr[offset*4 + 2] = value( m1, m2, h -120 );
    optr[offset*4 + 3] = 255;
}
```

```
__global__ void float_to_color( uchar4 *optr,
                                const float *outSrc ) {
    // map from threadIdx/BlockIdx to pixel position
    int x = threadIdx.x + blockIdx.x * blockDim.x;
    int y = threadIdx.y + blockIdx.y * blockDim.y;
    int offset = x + y * blockDim.x * gridDim.x;

    float l = outSrc[offset];
    float s = 1;
    int h = (180 + (int)(360.0f * outSrc[offset])) % 360;
    float m1, m2;

    if (l <= 0.5f)
        m2 = l * (1 + s);
    else
        m2 = l + s - l * s;
    m1 = 2 * l - m2;

    optr[offset].x = value( m1, m2, h+120 );
    optr[offset].y = value( m1, m2, h );
    optr[offset].z = value( m1, m2, h -120 );
    optr[offset].w = 255;
}


#if _WIN32
    //Windows threads.
    #include <windows.h>

    typedef HANDLE CUTThread;
    typedef unsigned (WINAPI *CUT_THREADROUTINE)(void *);

    #define CUT_THREADPROC unsigned WINAPI
    #define  CUT_THREADEND return 0

#else
    //POSIX threads.
    #include <pthread.h>

    typedef pthread_t CUTThread;
    typedef void *(*CUT_THREADROUTINE)(void *);

    #define CUT_THREADPROC void
    #define  CUT_THREADEND
#endif

//Create thread.
CUTThread start_thread( CUT_THREADROUTINE, void *data );

//Wait for thread to finish.
void end_thread( CUTThread thread );

//Destroy thread.
void destroy_thread( CUTThread thread );

//Wait for multiple threads.
void wait_for_threads( const CUTThread *threads, int num );

#if _WIN32
    //Create thread
    CUTThread start_thread(CUT_THREADROUTINE func, void *data){
        return CreateThread(NULL, 0, (LPTHREAD_START_ROUTINE)func, data, 0, NULL);
    }

    //Wait for thread to finish
    void end_thread(CUTThread thread){
        WaitForSingleObject(thread, INFINITE);
        CloseHandle(thread);
    }

    //Destroy thread
    void destroy_thread( CUTThread thread ){
        TerminateThread(thread, 0);
        CloseHandle(thread);
    }

    //Wait for multiple threads
    void wait_for_threads(const CUTThread * threads, int num){
        WaitForMultipleObjects(num, threads, true, INFINITE);

        for(int i = 0; i < num; i++)
            CloseHandle(threads[i]);
    }

#else
    //Create thread
    CUTThread start_thread(CUT_THREADROUTINE func, void * data){
        pthread_t thread;
        pthread_create(&thread, NULL, func, data);
        return thread;
    }

    //Wait for thread to finish
    void end_thread(CUTThread thread){
        pthread_join(thread, NULL);
    }

    //Destroy thread
    void destroy_thread( CUTThread thread ){
        pthread_cancel(thread);
    }

    //Wait for multiple threads
```

```
    void wait_for_threads(const CUTThread * threads, int num){
        for(int i = 0; i < num; i++)
            end_thread( threads[i] );
    }

#endif



#endif  // __BOOK_H__
```

edited Oct 18 at 11:12
UTF-8
1,769 1 10 26

answered Oct 18 at 9:15
Piotr Lenarczyk
1 1