ROOT Geant4 Seminar

Sho Nagao

2023 version

目次

このセミナーの目標

Linux System の基礎的な知識を学ぶ

解析等で実際に使用するソフトウェアの基本的な使い方を学ぶ

数値シミュレーションの代表的な手法であるモンテカルロ法について学ぶ

C、C++言語を頻繁に使いますが、気軽な気持ちで学びましょう

利用するもの

パソコン

（最低　CPU: Core-i5相当、Memory: 8 GB、Strage:128GB）
（推奨　CPU: Core-i7相当、Memory: 16 GB、Strage:256GB）

Windowsの方

ssh環境、X-window環境（VcXsrv）、Windows Subsystem for Linux (WSL v2)、Ubuntu

Linux系の方

特になし

Macの方

X-window環境（XQuartz）、Windows Subsystem for Linux (WSL v2)、Ubuntu

ROOT、Geant4の環境が手元にあるとなお良い

※ インストール方法もセミナー内でサポートします

※ Macの方は リモートPCに接続して Geant4 を動かしたとき、Viewer が上手に表示されない問題がありますのでローカルに持っておくこと推奨。ただし Mac のサポートは不可

資料等

資料、参考プログラムは GitHub上からダウンロード可

https://github.com/shonagao-nex/RootGeant-Seminar.git

直接アクセスするか、Cloneする

Clone方法

ターミナル上で

$ git clone https://github.com/shonagao-nex/RootGeant-Seminar.git

※ ダウンロードしてきたファイルのことを「リポジトリ上にあるファイル」と呼ぶことにします

※ GitHubアカウントを持っていない方は今すぐ作りましょう

※ Git の使い方も少しやります

記載のルール

コマンドを記載する場合、
　ユーザー権限で実行するコマンドは頭に「$」が、
　管理者 (root) 権限で実行するコマンドは頭に「#」もしくは sudo コマンドが、
　ROOT内で実行するものに関しては「root [0]」が
付きます

各自で変更が必要な個所はイタリックで記載します。

スペースが入るときには明示的に「␣」記号を入れるときがあります。

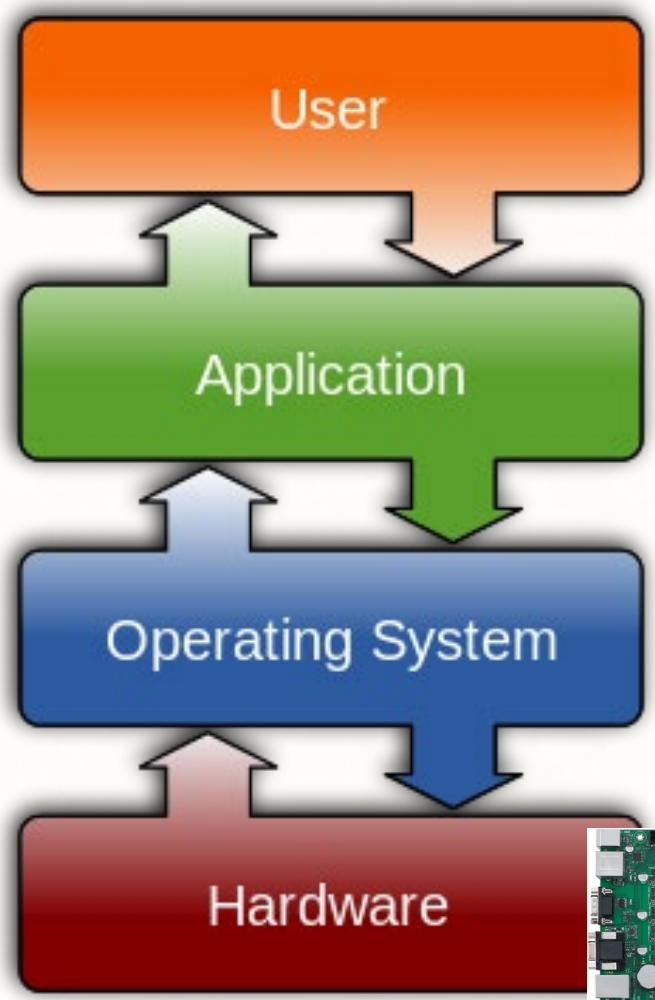
1. はじめに
2. Operating System
3. Windows
4. ROOT, Geant4 導入方法
5. Linux 環境設定
6. 鍵認証システム
7. バージョン管理システム
8. ROOTの使い方
9. モンテカルロ法
10. Geant4の使い方

パソコンの構成

ROOT、Geant4

Windows、Mac、Linux

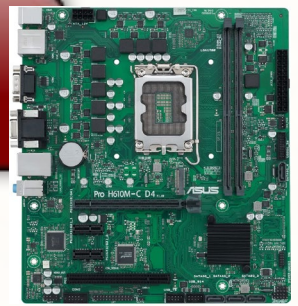
マザーボード、CPU、GPU、
メモリ、ストレージ、電源



プログラミング言語

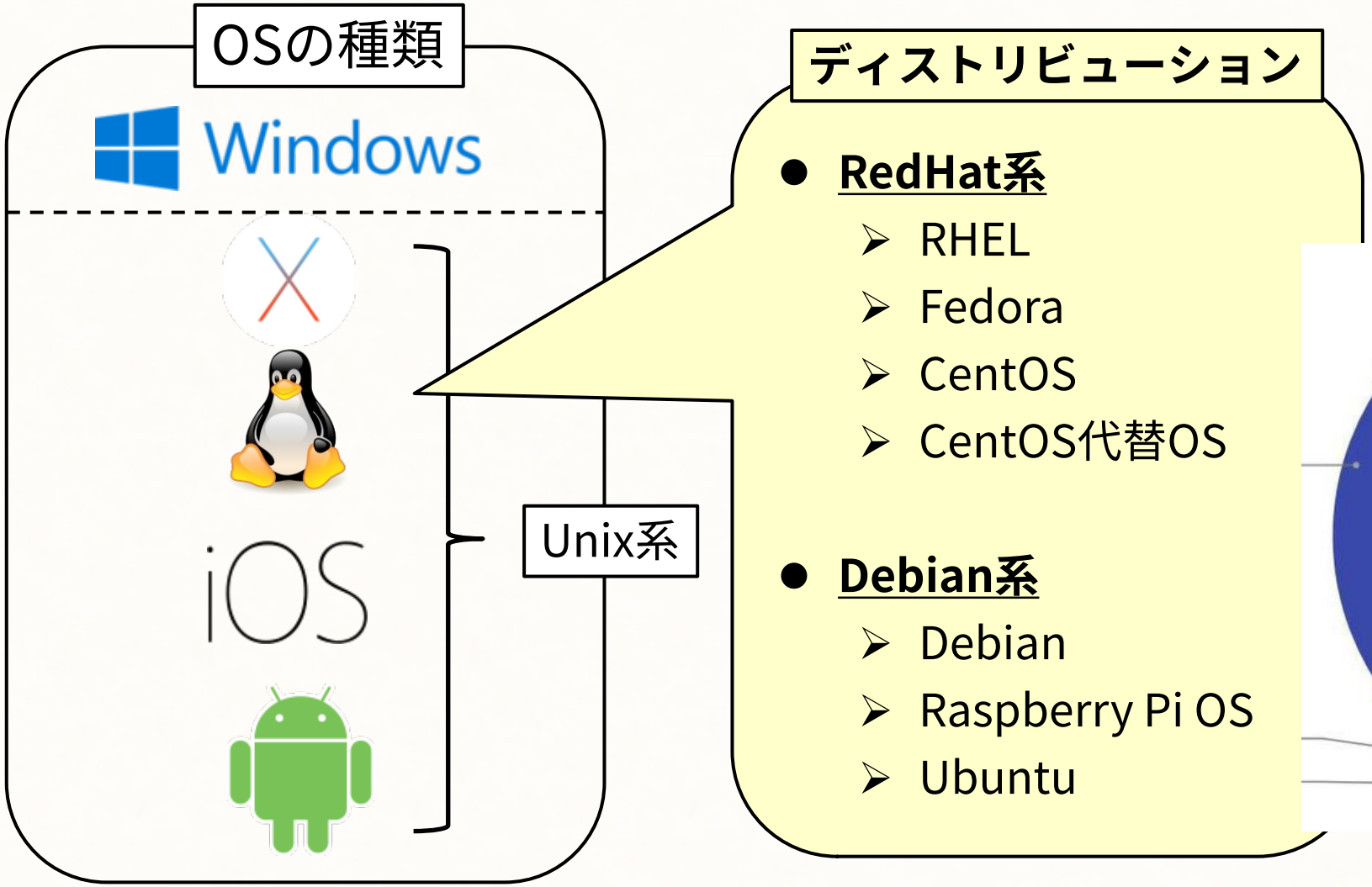
アセンブリ言語

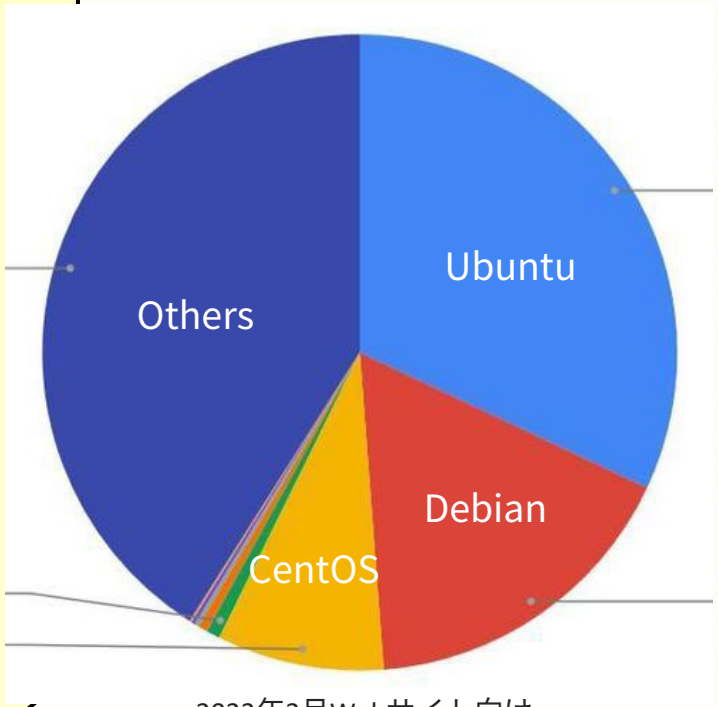
ビット情報



Operating System (OS)

ハードウェアを利用者に分かりやすいよう翻訳する機構





2023年2月Webサイト向け
Linuxディストリビューションシェア

1. はじめに
2. Operating System
3. Windows
4. ROOT, Geant4 導入方法
5. Linux 環境設定
6. 鍵認証システム
7. バージョン管理システム
8. ROOTの使い方
9. モンテカルロ法
10. Geant4の使い方

Windowsの環境構築

別端末からLinux端末で作業するためには、コマンド入力するための ssh 環境と、画像をコントロールする X window system が必要。

Linux OS では両方環境が整っているが、Windowsでは環境の構築が必要。

ssh環境（上から簡単かつ軽量）

PowerShell：Windows10で軽量ならコレ。あまり困らない。

Putty, TeraTerm：以前までの定番。現状PowerShellがあれば使わないかも。

Cygwin：以前までの定番のひとつ。今やWSLの下位互換。

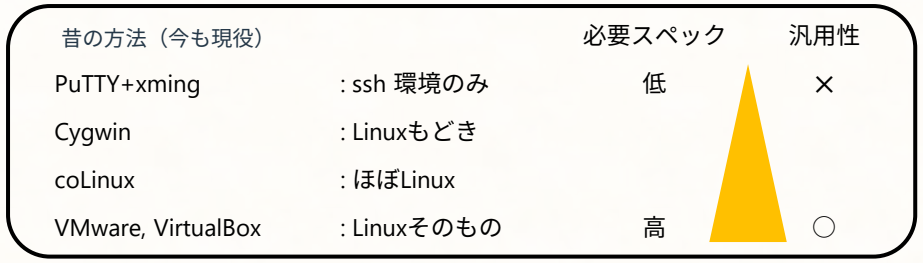
WSL：Windows Subsystem for Linux。ほぼLinux環境。OSの選択肢少ない、根っこの部分はいじれないなどの制限もある。

VMware：フルLinuxを使いたい場合はコレ。これさえあれば何でもできる。

X window環境

Xming：以前までの定番。今は、VcXsrvがお勧め。

VcXsrv：無償で全機能が使える。Xmingの完全上位互換。



ここでは、Windows Terminal + WSL + Ubuntu + VcXsrv を使った方法を紹介

VcXsrv環境構築

VcXsrvとは
VcXsrvのダウンロード
VcXsrvのインストール
VcXsrvの起動・設定

VcXsrv とは、Windows 上で動作する X window サーバー。

Xming が良く使われてきたが、更新が途絶えている、速度が遅いなどの問題を孕んでいる。

VcXsrv と ssh環境を組み合わせることで、他のPCからの画像を飛ばすことが可能となる。

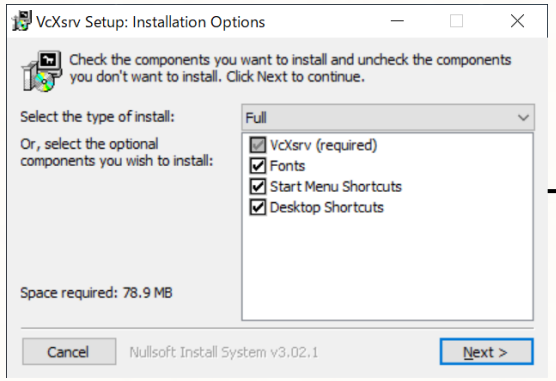
https://sourceforge.net/projects/vcxsrv/

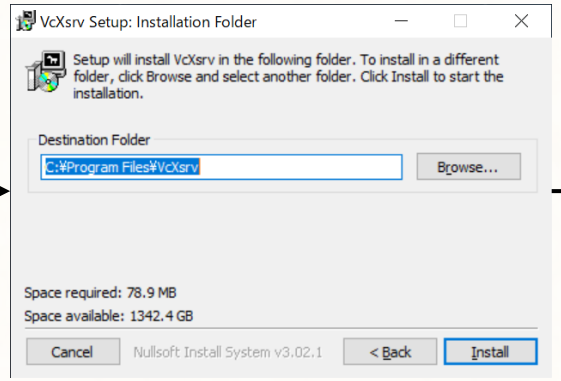
からダウンロード可能（リンクが生きていれば）。

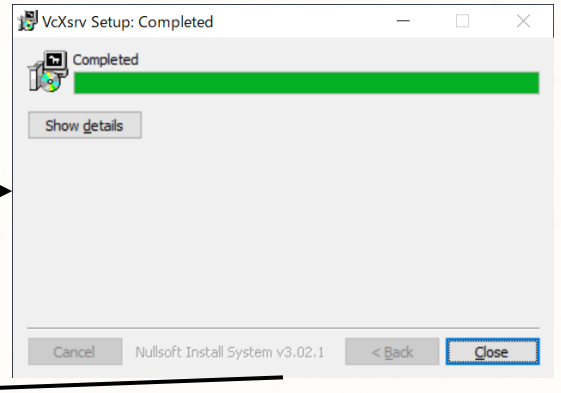
ダウンロードしたものをダブルクリックしてインストールすればよい。

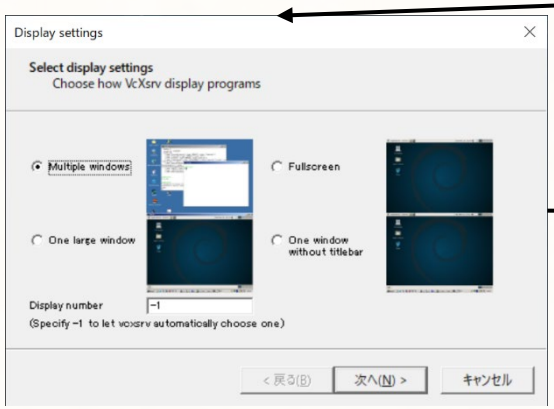
VcXsrv環境構築

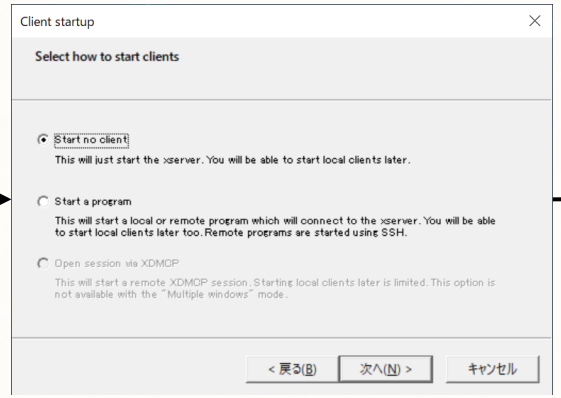
- VcXsrvとは
- VcXsrvのダウンロード
- VcXsrvのインストール
- VcXsrvの起動・設定



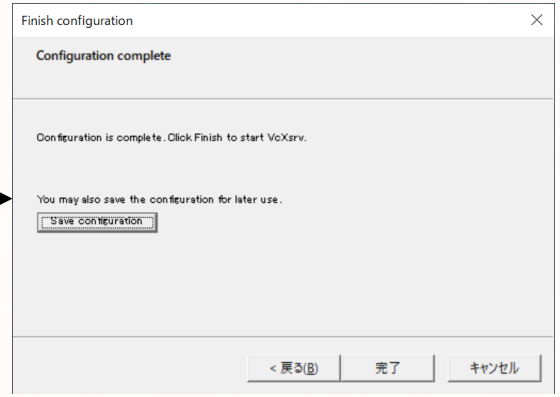








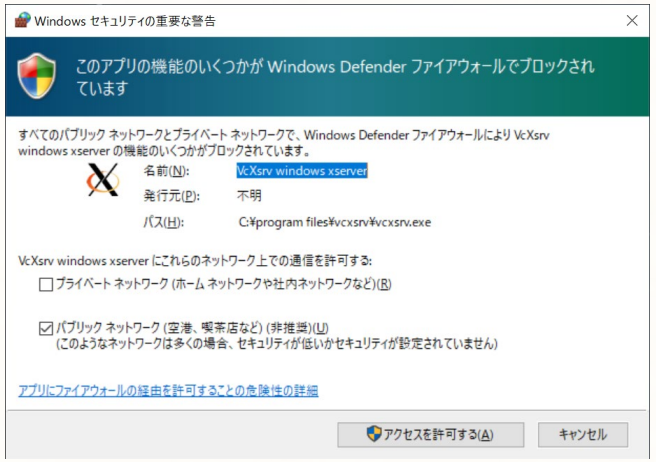
Native Opengl を OFF
Disable Access Control を ON に



ほぼデフォルトで良いが、OpenGLとAccess Control設定は忘れずに。
Configurationファイルを保存し、スタートアップ登録すれば、自動起動できる。

VcXsrv環境構築

- VcXsrvとは
- VcXsrvのダウンロード
- VcXsrvのインストール
- VcXsrvの起動・設定



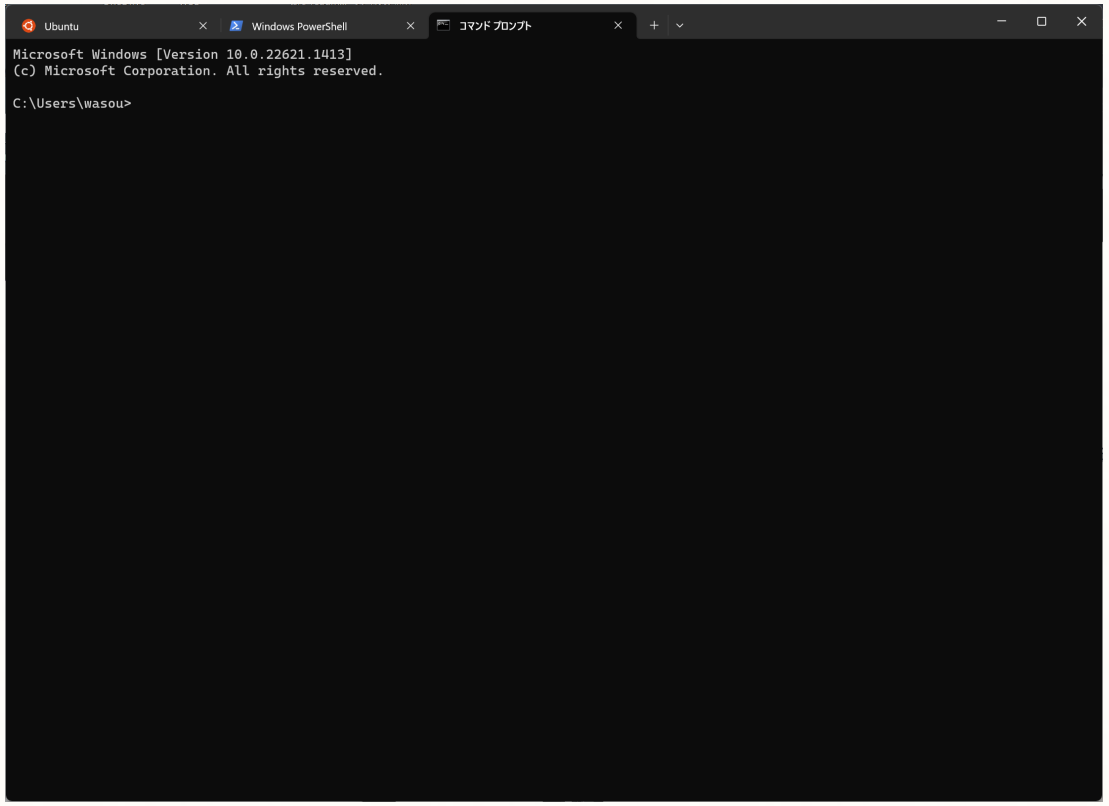
この画面が出たら「アクセスを許可する」をクリック
状況次第でプライベートネットワークも許可しておく

デスクトップにこのようなアイコンが

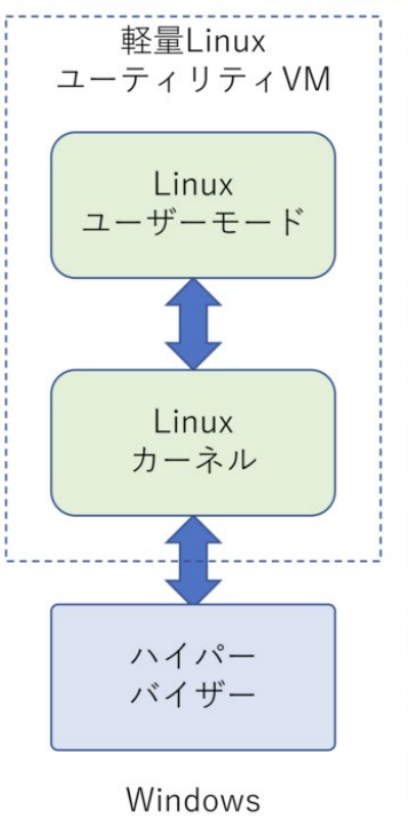
Windowsキー＋Rをクリックして「ファイル名を指定して実行」を立ち上げる
shell:startup　と入力
スタートアップフォルダが立ち上がるので、そこに config.xlaunch を移動（自動起動させるため）

Windows Terminal の準備

Linux の Terminal のように端末として利用可能
同一端末内で Ubuntu、PowerShell、コマンドプロンプトなどを起動可能（便利）
ターミナルの外観も良い感じ
Microsoft Store からインストール可能



WSL2の準備 (Windows Subsystem for Linux 2)



Windows 10 バージョン 2004 以上 (ビルド 19041 以上) または Windows 11 を実行している必要あり

それ以前（x64 システムの場合:バージョン 1903 以降、ビルド 18362 以上。ARM64 システムの場合:バージョン 2004 以降、ビルド 19041 以上。）も利用可能だが、インストール方法がやや煩雑なのでここではパス

１．管理者として PowerShell を開き、以下を実行

　　wsl --install

２．インストールが完了したら、PCを再起動

３．再起動後、自動でUbuntuが起動する（はず）のでユーザ名とパスワードを設定

４．正しくインストールできていれば、PowerShell 上から以下のように確認できる

```
PS C:\Users\wasou> wsl -l -v
  NAME      STATE           VERSION
* Ubuntu    Running         2
```

Tips

WSL上のUbuntuとWindows間でファイルのやり取りが可能

デフォルトでは Ubuntu のホームディレクトリは以下の場所に生成されている

¥¥wsl.localhost¥Ubuntu¥home

また、コマンドが重複するので、設定→操作から コピーを「ctrl+shift+c」ペーストを「ctrl+shift+v」としておくのがおススメ

※　インストールできない場合は「Windows の機能の有効化または無効化」を開き「Linux 用 Windows サブシステム」にチェックが入っているか確認

1. はじめに
2. Operating System
3. Windows
4. ROOT, Geant4 導入方法
5. Linux 環境設定
6. 鍵認証システム
7. バージョン管理システム
8. ROOTの使い方
9. モンテカルロ法
10. Geant4の使い方

WSLに色々インストール

Windows Terminal を起動して WSLを起動
以下の手順でOSアップデート、ソフトウェアインストール例

アップデート（定期的に）

```
$ sudo apt update
$ sudo apt upgrade
```

インストール

```
$ sudo apt install cmake g++ libx11-dev libxpm-dev libxft-dev libxext-dev libxmu-dev
$ sudo apt install gfortran libgif-dev libtiff-dev libjpeg-dev libxml2-dev libfftw3-dev
$ sudo apt install libgsl-dev cmake-curses-gui xxHash
$ sudo apt install libxerces-c-dev qtbase5-dev libqt53drender5 libmotif-dev
```

Python関係

```
$ sudo apt install python3 python3-dev pip3
```

Python3.9を導入したい場合

```
$ sudo apt install software-properties-common
$ sudo apt install python3.9 python3.9-dev pip3.9
$ sudo pip3.9 install numpy scipy matplotlib jupyter pandas
```

WSLにROOTをインストール

インストール

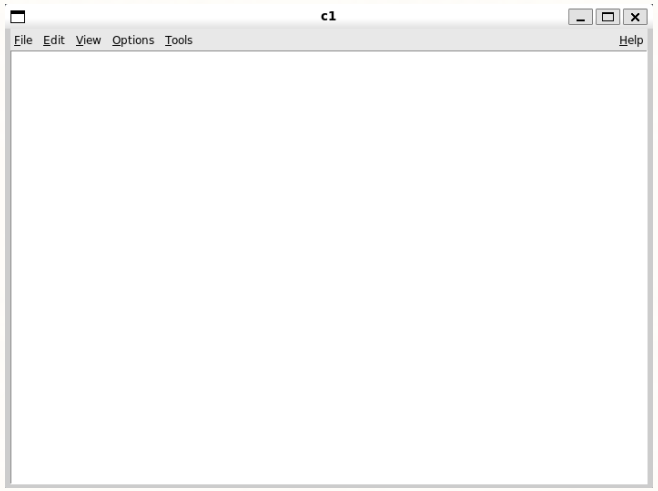
$ cd ~/Download
$ wget https://root.cern/download/root_v6.24.08.source.tar.gz; tar zxvf root_v6.24.08.source.tar.gz
$ mkdir root_v6.24.08_build; cd root_v6.24.08_build
$ sudo mkdir /usr/local/share/applications/root_v6.24.08
$ cmake ../root-6.24.08/
$ ccmake .
　CMAKE_INSTALL_PREFIX　/usr/local/share/applications/root_v6.22.08
　fftw3, fortran, minimal, minuit2, pyroot, roofit あたりがONになっているか確認
　"c"でコンパイル、"g"でMakefile生成、エラーが出る場合はパッケージ不足している可能性
$ make -j4（4-threads でコンパイル、数字は自分のPCのスペックと相談）
$ sudo make install

環境設定

~/.bashrc に以下を追記
　export ROOTSYS=/usr/local/share/applications/root_v6.22.08
　source $ROOTSYS/bin/thisroot.sh
$ source ~/.bashrc で設定を反映

起動

$ root
root[0] gROOT->GetVersion()　としてインストールしたROOTのバージョンが出れば成功
root[0] TCanvas *c1 = new TCanvas()　として右のような画面が出てきたら成功
root[0] .q　としてROOTを出る



WSLにGeant4をインストール

インストール

```
$ cd ~/Downloads
$ wget https://gitlab.cern.ch/geant4/geant4/-/archive/v10.7.4/geant4-v10.7.4.tar.gz;  tar zxvf geant4.10.07.p04.tar.gz
$ mkdir geant4.10.07.p04_build;  cd geant4.10.07.p04_build
$ sudo mkdir /usr/local/share/applications/geant4.10.07.p04
$ cmake ../geant4.10.07.p04/
$ ccmake .
        CMAKE_INSTALL_PREFIX          /usr/local/share/applications/geant4.10.07.p04
        GEANT4_BUILD_MULTITHREADED    ON
        GEANT4_INSTALL_DATA           ON
        GEANT4_USE_GDML               ON
        GEANT4_USE_OPENGL_X11         ON
        GEANT4_USE_QT                 ON
        GEANT4_USE_RAYTRACER_X11      ON
        GEANT4_USE_SYSTEM_EXPAT       ON
        GEANT4_USE_XM                 ON

        "c"でコンパイル、"g"でMakefile生成、エラーが出る場合はパッケージ不足している可能性
$ make -j4;  sudo make install
```

環境設定

```
~/.bashrc に以下を追記
        export G4INSTALL=/usr/local/share/applications/geant4.10.07.p04
        source $G4INSTALL/bin/geant4.sh
$ source ~/.bashrc で設定を反映
```

1. はじめに
2. Operating System
3. Windows
4. ROOT, Geant4 導入方法
5. Linux 環境設定
6. 鍵認証システム
7. バージョン管理システム
8. ROOTの使い方
9. モンテカルロ法
10. Geant4の使い方

Linux 環境設定

自分独自の環境がある人はパス

```
$ cd ~
$ git clone https://github.com/shonagao-nex/RootGeant-Seminar.git
```

リンクを貼る

```
$ ln -s  RootGeant-Seminar/environments/bashrc  .bashrc
```

inputrc, colorrc, rootlogon.C, rootkinematics.C, rootmacro.C, vim, vimrc も同様

設定反映

```
$ source  .bashrc
```

※　色設定はダークモードでの利用を想定しているので適宜変更

環境変数設定ファイルコメント

今回は bash を利用（利用シェアが圧倒的に高く、情報量が多いため）。
csh, tcsh, zsh 等々を利用しても良い

```
HISTCONTROL=ignoreboth
export PROMPT_COMMAND="history -a; history -c; history -r; $PROMPT_COMMAND"
shopt -u histappend
export HISTTIMEFORMAT='%F %T '
export HISTSIZE=10000
export HISTFILESIZE=20000
```

```
### ROOT ###
export ROOTSYS=/usr/local/share/applications/root_v6.24.08
source $ROOTSYS/bin/thisroot.sh
### GEANT4 ###
export G4INSTALL=/usr/local/share/applications/geant4.10.7.4
source $G4INSTALL/bin/geant4.sh
```

```
eval `dircolors -b ~/.colorrc`
alias ls='ls -CF --color=auto'
alias cp='cp -i'
alias mv='mv -i'
alias rm='rm -i'
alias cd='cd -P'
alias vi='vim'
alias root='root -l'
alias clean='rm -f *~'
```

強く推奨

1. はじめに
2. Operating System
3. Windows
4. ROOT, Geant4 導入方法
5. Linux 環境設定
6. 鍵認証システム
7. バージョン管理システム
8. ROOTの使い方
9. モンテカルロ法
10. Geant4の使い方

鍵認証システム

リモートマシンへアクセスする場合、鍵認証システムを利用することを推奨

公開鍵：暗号化専用の鍵 (id_rsa.pub)　ホストが製作、クライアントが使う
秘密鍵：復号専用の鍵 (id_rsa)　　ホストが製作、ホストが死守

lambda から farm へ鍵認証を使ってログインする場合、
クライアント：lambda　　サーバー：farm

鍵の作成および設定方法

```
$ cd ~/.ssh
$ ssh-keygen  -t  rsa                          (秘密鍵、公開鍵の作成)
      Enter file in which to save the key (/home/username/.ssh/id_rsa):   ← Enter
      Created directory '/home/username/.ssh'.                            ← Enter
      Enter passphrase (empty for no passphrase):                         ← パスフレーズを入力（大文字小文字数字等々を織り交ぜて）
      Enter same passphrase again:                                        ← もう一度パスフレーズを入力
$ chmod  600  id_rsa                           (鍵のアクセス権設定)
$ scp  id_rsa.pub  farm:~/.ssh/                (公開鍵を転送)
$ cd ~/.ssh
$ cat id_rsa.pub >>  authorized_keys           (公開鍵を登録)
$ chmod  600  authorized_keys                  (公開鍵のアクセス権設定)
```

使い方

```
$ ssh ␣ -Yi ␣.ssh/id_rsa ␣ username@farm
```

鍵認証の補足

より便利に使うため、クライアント側で以下の操作を行う

```
$ cd ~/.ssh
```

config ファイルを作成、編集

```
Host farm
    hostname farmのIP Address（サーバー上、/sbin/ifconfig で確認可能）
    identityfile ~/.ssh/id_rsa
    user username
```

```
$ chmod 600 config          （configの権限設定）
```

~/.bashrc に以下を追加

```
alias farm 'ssh –Y farm'
$ source .bashrc            （環境変数を設定）
$ eval `ssh-agent`          （ssh-agentを起動）
$ ssh-add ~/.ssh/id_rsa     （ssh-agentに鍵を登録）
$ farm                      （ログイン）
```

ssh-agent を常に起動したいのであればその旨を環境変数に追記すればよい。

ファイルコピー

Unix系の場合

scp コマンド

$ scp コピー元 コピー先

$ scp nagao@lambda.phys.tohoku.ac.jp:~/test.txt ./

(Lambdaホームディレクトリの test.txt を 自分のPCの今のディレクトリにコピー)

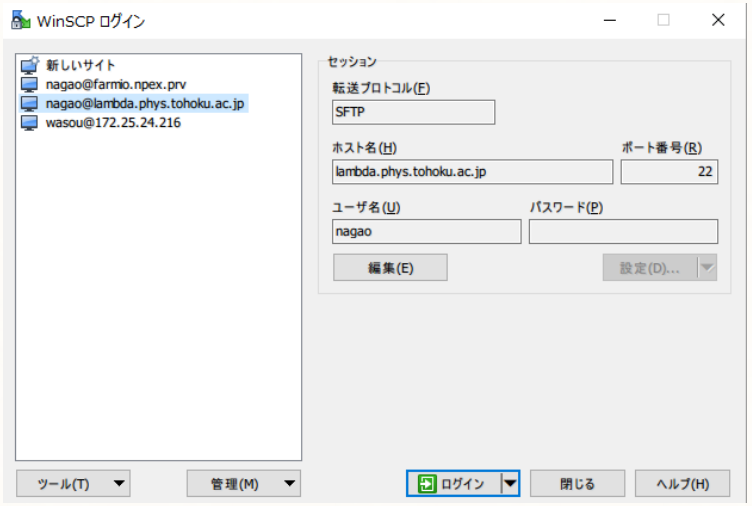
rsync コマンド

$ rsync -auv --progress nagao@lambda.phys.tohoku.ac.jp:~/test.txt ./

Windowsの場合

WSLでSCPコマンド等々利用可能だが、専用ソフト（WinSCP）も便利

https://winscp.net/eng/docs/lang:jp にアクセスしてダウンロード・インストール



1．ホスト名を入力

2．ユーザー名を入力

3．保存（ログインの隣？）して

4．ログインをクリック

5．パスワードを入力

1. はじめに
2. Operating System
3. Windows
4. ROOT, Geant4 導入方法
5. Linux 環境設定
6. 鍵認証システム
7. バージョン管理システム
8. ROOTの使い方
9. モンテカルロ法
10. Geant4の使い方

ブランチを切ってローカルリポジトリにコミット
リモートリポジトリにプッシュしてマージする
を理解する

GitHub

GitHubとは、Gitを利用した、開発者を支援するWebサービスです。
自分でサーバーを立ててリポジトリを作成する必要なし。
リポジトリを public にすれば、他人とコードの共有も可能です。
執筆中の論文の管理にも使えます。
開発のレビューなども簡単にできるので非常に便利。是非アカウントを作っておきましょう。

バージョン管理システムとは

バージョン管理システム（version control system）とは、コードなどの編集履歴を管理、保管するシステム

システム管理サーバー
（リポジトリ）

作業者は、基本的に以下の流れで作業を行う。

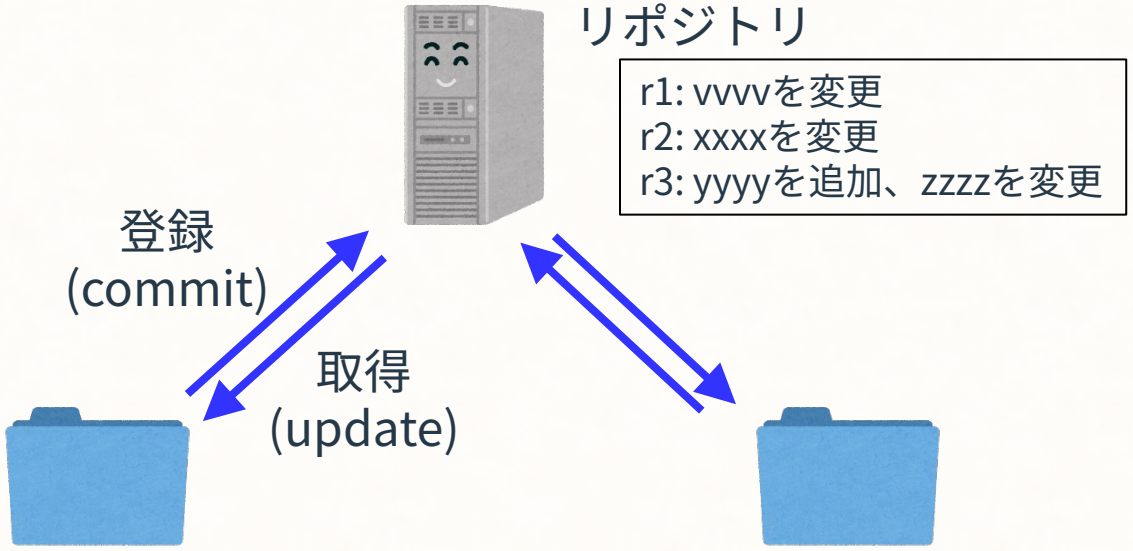
1．リポジトリをローカルに複製（clone）する。
2．ローカルで修正・追加（add）・削除（rm）を行う。
3．変更内容をリポジトリに反映させる（commit）。

作業者
（ローカル）

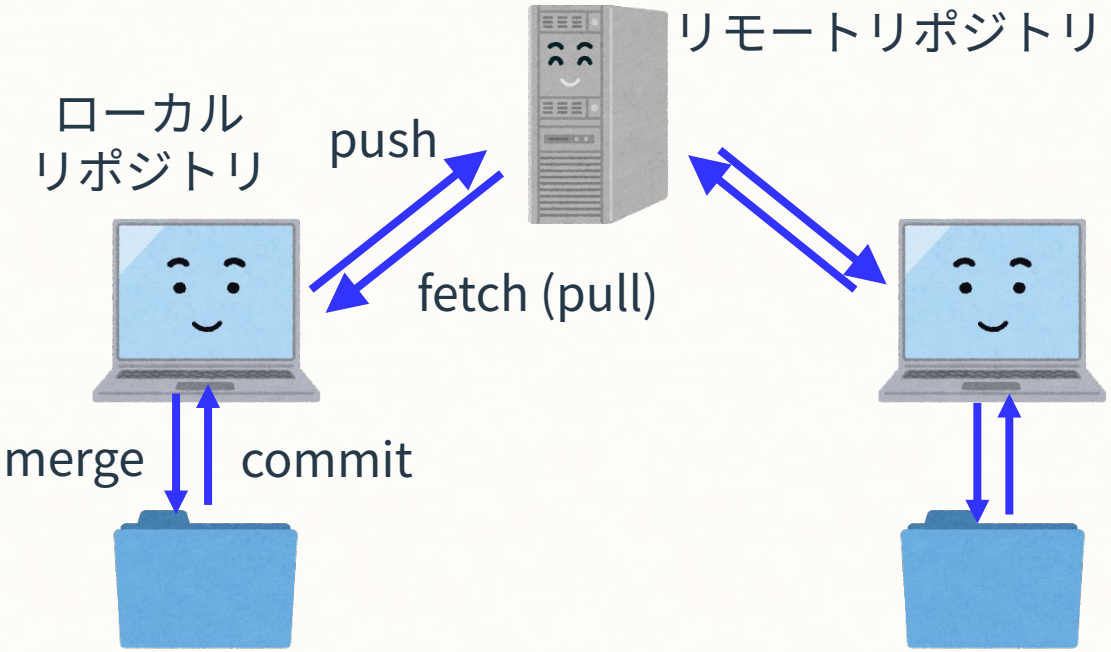
バージョン管理システムの種類

バージョン管理システムは「集中型」「分散型」に分けられる。

集中型：バージョン管理を一か所のリポジトリがリビジョン番号を割り振って行う。CVS、Subversion



分散型：各作業者がリポジトリを保有している（ローカルリポジトリ）。Git、Mercurial



集中型の方が分かりやすいが、作業者は他の作業者の内容がリポジトリに反映されるまで待つ必要がある。

近年は分散型の Git が頻繁に使われる。

作業者の流れ（Subversion）

リポジトリ名は vcs。
リポジトリに既にある test.cc を変更、新たに hoge.cc を追加して登録することを考える。

1. svn checkout vcs	リポジトリの内容を複製
2. 内容を編集、追加	
3. svn update	最新の状態にする
4. svn add hoge.cc	ファイルを追加
5. svn commit test.cc hoge.cc -m “modify test.cc, add hoge.cc”	作業内容をリポジトリに反映

※一度チェックアウトしてしまえば、次からは手順１は必要ない。

その他

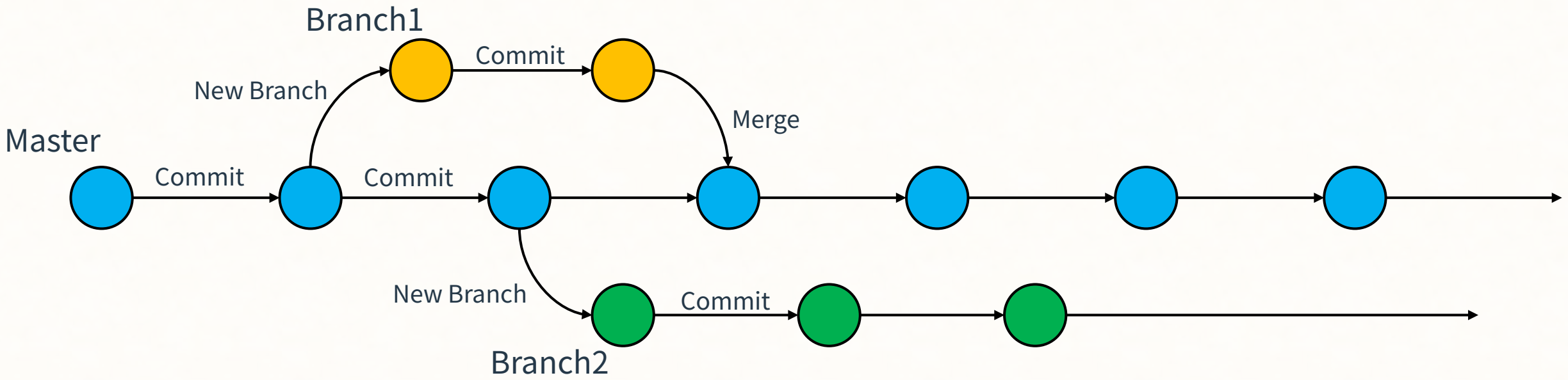
変更ログを確認したい	svn log
リポジトリとの差を確認したい	svn diff test.cc
現在の状況を確認したい	svn status
変更内容をリポジトリの内容に戻したい	svn revert test.cc

ブランチという考え方（Git）

Git はブランチという機能が充実している。他者の変更内容と競合を避けるため、作業者は、

1. Master の内容を Branch を切って編集作業を進め
2. ローカルリポジトリに変更内容を逐次 commit
3. ある程度形になったところで Branch のリモートリポジトリに push し
4. プルリクエストを作成
5. 管理者が Master に Branch の内容を merge する。

といったように進めることが多い。Subversionでも似たような機能があるが充実していない。
開発方針によっては、Master のみで進めることもある。



作業者の流れ（Git）

リポジトリ名は vcs。master ブランチから local ブランチを作成。リポジトリに既にある test.cc を変更、新たに hoge.cc を追加。最終的に master ブランチにマージすることを考える。

1. git clone vcs	リモートリポジトリの内容を複製
2. git pull	最新の状態にする
3. git branch local	local ブランチを作成
4. git checkout local	local ブランチに移動
5. 内容を編集、追加	
6. git add hoge.cc	ファイルを追加
7. git commit test.cc hoge.cc -m “modify test.cc, add hoge.cc”	作業内容をローカルリポジトリに反映
8. git push origin local	ローカルをリモートに反映
9. git checkout master	master ブランチに移動
10. git pull origin master	master ブランチを最新の状態にする
11. git merge local	local を master にマージ
12. git push origin master	内容をリモートに反映

※一度クローンしてしまえば、次からは手順１は必要ない。

※ブランチ機能を使わないのであれば灰色は不要。

Subversion と Git のコマンド

内容	Subversion	Git
リポジトリを作成	svn create	git init
リモートリポジトリの内容を複製	svn checkout	git clone
作業ディレクトリを更新	svn update	git pull
ローカルリポジトリのみ更新		git fetch
ローカルリポジトリの内容を作業ディレクトリに反映		git merge
ファイルの追加（移動、削除）	svn add (mv, rm)	git add (mv, rm)
変更をリポジトリに反映	svn commit	git commit
変更をリモートリポジトリに反映		git push
変更の取消	svn revert	git reset
状態確認	svn status	git status
差分確認	svn diff	git diff
ログの確認	svn log	git log
ブランチの作成、一覧		git branch
ブランチの切替		git checkout
ブランチのマージ	svn merge	git merge

1. はじめに
2. Operating System
3. Windows
4. ROOT, Geant4 導入方法
5. Linux 環境設定
6. 鍵認証システム
7. バージョン管理システム
8. ROOTの使い方
9. モンテカルロ法
10. Geant4の使い方