ネットワーク
システム管理 #07

たかさきこうや

1

先週のおさらい

- UNIXで良く用いるコマンド
- ファイルやディレクトリの操作、移動

2

ファイル編集

- Echoでリダイレクトをして
  ファイルを作成するのは大変だ
- エディタはないのか？
- ※エディタ＝Windowsで言うところのメモ帳のようなもの
- ※ワープロソフトではない

3

UNIX上のテキストエディタ

- EmacsやViというエディタが存在する
- UNIX上の定番エディタ
  - Windowsにも移植されている
  - 操作性は独特だけど、慣れれば便利
- Xyzzyというエディタも、emacsを参考に
  作られている

4

豆知識

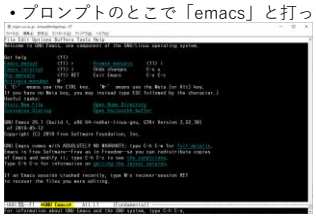
- emacs系のエディタ
  - emacs　/ mule / xemacs
- vi系のエディタ
  - vi / vim / nvi

- この２つのエディタは「ネットワーク」業界や
  「システム」業界ではよく使われる
  - それ系の仕事をするなら覚えておいて損はない

5

コマンド:emacs

- TeraTermでログイン
- プロンプトのとこで「emacs」と打ってEnter.



なんかゴチャゴチャ
出るけど、起動時の
コピーライト表記とか
なので気にしないでよし

6

まず思うこと

- 「うわー。どうやって使うんだろう…」
- 基本は、Ctrl-なにか　というキー操作をする
  - Windowsで言うところの「ショートカットキー」
- エディタの終了は
  - Ctrl-x Ctrl-c
    - コントロールキーを押しながらxとcを連続で押す
  - まずは終わらせ方からマスターしよう

7

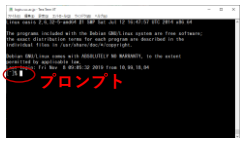
ここで再度注意

- emacsは現時点で、sshで接続した
  「login.cuc.ac.jp」サーバの上で動いている
- TeraTermを終了させても、それは
  「サーバとの接続が切れるだけ」であり、
  「**サーバ上で動いているemacsが終了したこと**」
  には**ならない**ので注意

8

見分け方

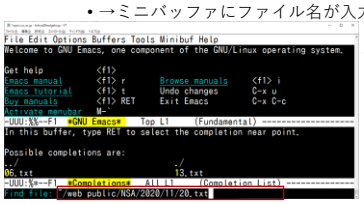
- ミニバッファがあれば
  「emacs起動中、の状態」
- ミニバッファがなければ
  「コマンド入力待機中、の状態」



9

では、作業をしてみよう

- ファイルの呼び出し（新規作成）
  - Ctrl-x Ctrl-f
    - →ミニバッファにファイル名が入力可能になる



- ファイルが既に存在していれば、そのファイルを開く
- 存在していなければ
  新規に作成

10

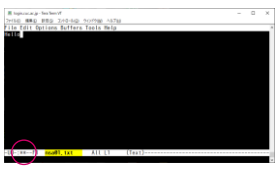
作成するファイル

- web_public/NSA/2023/12/01.txt
  - 勿論、まずは12というディレクトリを2023の下に作るんやで…
- 作成したら、
  「Good morning!
  My name is ●●●.」
  と書いてください(●●●は自分の名前)

11

適当に編集してみる

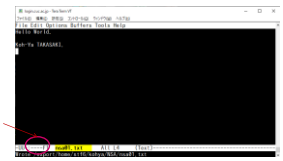
- ミニバッファでファイル名を指定
  - →メインバッファに入力が可能になる
- 編集して保存していない場合、
  メニューバーに
  「**」と表示される



12

編集したら保存

- 編集したファイルの保存
  - Ctrl-x Ctrl-w
    - 名前を決めずに編集を開始した場合は、これで保存する
      ファイル名を聞いてくる
- 上書き保存
  - Ctrl-x Ctrl-s
    - 保存するファイル名は
      聞いてこない



13

困ったときのキー操作

- Ctrl-l
  - 再描画（表示が乱れた時用）
    - 1回押すとカーソルを画面の中心にして再描画
    - 2回押すとカーソルを画面の最上部にして再描画
- Ctrl-g
  - やりかけた操作をキャンセル
- Ctrl-x u
  - 直前の操作をアンドゥ

14

「切り取り」「コピー」「貼り付け」

- Ctrl-Space
  - マークセット(ミニバッファに Mark Setと出る)
- Esc-w
  - コピー(Ctrl-Spaceの場所から今の場所まで)
- Ctrl-w
  - 切り取り
- Ctrl-y
  - 貼り付け

15

ページ、カーソルの移動(参考)

- Ctrl-e
  - 行終端まで移動
- Ctrl-a
  - 行始端まで移動
- Esc-<
  - 文書の初めまで移動
- Esc->
  - 文書の終わりまで移動

16

検索と置換(参考)

- Ctrl-s
  - 次の候補をサーチ
- Esc-%
  - 置換

17

削除(参考)

- Ctrl-h カーソルの直前の文字を削除
- Ctrl-d カーソルの直後の文字を削除
- Ctrl-k 一行削除

18

特殊な操作(参考)

- Ctrl-x i 別のファイルをインサート
- Ctrl-t 直前の1文字とカーソルのある文字を入れ替え

19

バッファ操作

- バッファ＝編集可能な対象
- emacsは複数のバッファを持てる
  - メモ帳だって、複数起動できるでしょ？
- また、バッファに出ていない
  （非表示なファイル）というのも存在する

20

バッファ分割、バッファを閉じる

- Ctrl-x 2
  - バッファを縦2画面に分割
- Ctrl-x 3
  - バッファを縦2画面に分割
- Ctrl-x k
  - カーソルのあるバッファのファイルを閉じる
- Ctrl-x o
  - バッファ間を移動

21

バッファの切り替え

- Ctrl-x 0
  - 分割したバッファのうち、カーソルのある方を非表示(ファイルは閉じない)
- Ctrl-x 1
  - 分割したバッファのうち、カーソルのあるもの以外を非表示(ファイルは閉じない)
- Ctrl-x b
  - バッファに表示されていない（裏の）ファイルに
    切り替える

22

そういうコマンド覚えないと駄目？

- 駄目というわけではないけど、Photoshopでも
  なんでもキーボードショートカットを覚えると作業早いよ
- Windowsのアプリケーションも同じ
- キーボードから手を離してマウスに手を持っていくのは
  それ自身が時間のロスだったりするので…
  - というか、みんな、タッチタイピングぐらい
    出来るようになってるよね？

23

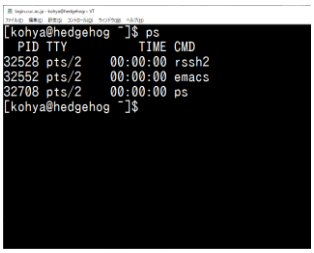
サスペンド

- Ctrl-z
  - 一旦emacs自身をサスペンドする
  - プログラムを一時停止
    - 終了とは違うので、再開させれば編集中のファイルが呼び出せる
  - コマンドプロンプトのところでfgと打てば、
    emacsを再開できる

24

コマンド:ps

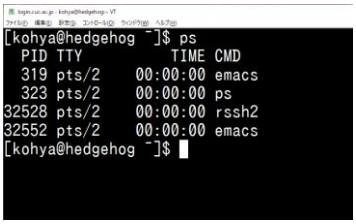
- psコマンドを実行して emacsが居れば、 「今、中断している プログラムが居そう」な ことがわかる



25

コマンド:jobs

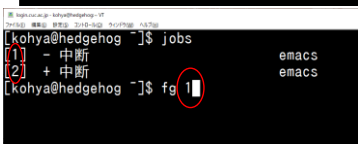
- もし、Ctrl-zで止めた emacsが「ふたつ」 あるなら、どちらかを 選んで再開しないと いけない



26

コマンド:jobs

- jobsコマンドで確認すると、二つ「中断」しているのがわかる
- fg ジョブ番号 で 任意の「中断中の プログラム」を再開 させられる



27

viというエディタもある

- 今までのはEmacsの操作方法だけど、viというエディタもある
  - 編集モードとコマンドモードを行き来するという、やや なじみの薄い概念を持っている
- % vi ファイル名 で起動
  - この状態だと「コマンドモード」なので、カーソルの移動や コピペ、置換などはできるが「文章の入力そのものはできない」
  - i キーを押すと 「編集モードになる」が、この状態では カーソルの移動ぐらいしかできない
  - Escキーを押すと「コマンドモード」に戻る

28

viの終了方法

- 保存するときは「:w」と打つ
- 終了するときは「:q」(保存して終了したければ:wq)
- 保存しないで終了したいときは「:q!」

- Emacsが入っていないシステムでも、viは入っているという UNIXも多数あるので、簡単な編集ぐらいはできても損はない

29

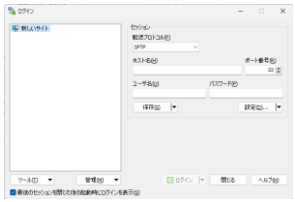
では試しに

- web_public/NSA/2023/12/02.txt というファイルを作る
- 中に、 「今日12月02日は土曜日だ。CUCはお休みだ。 何して過ごそうか? 楽しみだなあ!」 と書いて保存してください
  - ただし、赤字の部分は半角英数で

30

この状態で

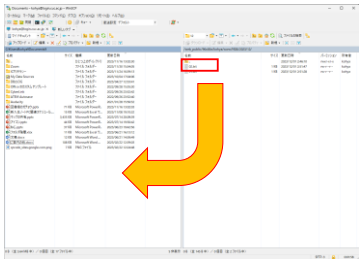
- 初回にインストールした「WinSCP」を起動
  - リモートとローカルのサーバの間で、ファイルを転送するアプリケーション
- login.cuc.ac.jp にWinSCPで接続し、今作ったファイルを手元のPCにダウンロードする
  - ホスト名:login.cuc.ac.jp
  - ユーザ名とパスワードはいつものやつ
  - 情報は保存しても良い



31

作ったNSA/2023/12/02.txtを

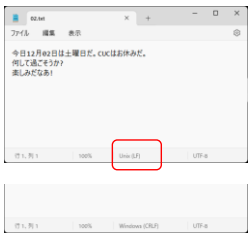
- 自分のPCの「ドキュメント」か「デスクトップ」にコピー
- そのうえで、Windowsのメモ帳で該当のファイルを開いてみる



32

すると、普段PCでメモ帳で

- ファイルを作った時と若干様子が違う…
- Unix(LF)となる
- ちなみに普段はWindows(CRLF)となるはず



33

なにこれ？

- 以前に、OSによって使われる文字コードは違うという話はした
  - 最近はみんなUnicodeを使ってるよ、とも言った
    - UTF-8かUTF-16を使っているよ、とも
- でも、実はOSによっては、「改行」に使われる特殊記号が違う

34

改行ってなに？

- タイプライターに起源があるんだけれど…
- 「改行」と一言で言っても、実は
  「次の行へ行く」
  「行の先頭へ行く」
  の二つの概念が組み合わさっている
- 「次の行へ行くこと」は
  「ラインフィード(Line Feed)」
- 「行の先頭へ行くこと」は
  「キャリッジリターン(Carriage Return)」

35

厄介なことに

- UNIXは、「LF」だけで改行とみなしている
- Macは、「CR」だけで改行とみなしている
- Windowsは「CR+LF」で改行とみなす

- というルールになっている
- なので、Windowsで作ったファイルをUNIXに持っていくと「CR」が余計になる
- UNIXで作ったファイルをWindowsに持ってくると「CR」が足らなくなる

36

そうしたらemacsを終了し

- ターミナルで以下のコマンドを実行
- cat 02.txt | sed -e 's/土/日/' | sed -e 's/02/03/' > 03.txt
- ちなみに、
- catコマンドは「ファイルの中身を全部表示」
- sedコマンドは「文字列の一部を置換する」

37

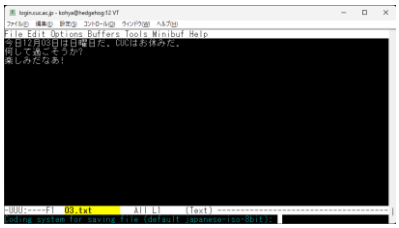
自信がないなら

- cat 02.txt
- cat 02.txt | sed -e 's/土/日/'
- cat 02.txt | sed -e 's/土/日/' | sed -e 's/02/03/'
- cat 02.txt | sed -e 's/土/日/' | sed -e 's/02/03/' > 03.txt
- という感じに順を追って処理を増やしていってあげるとよい

38

03.txtができたら、emacsで

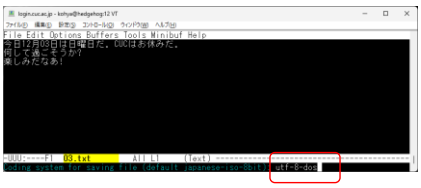
- そのファイルを開く
- そのうえで、「Ctrl-x Ctrl-m f」と入力する
  - Ctrlキーを押しながら x m と入力しCtrlキーを離して f



39

Coding system for saving file

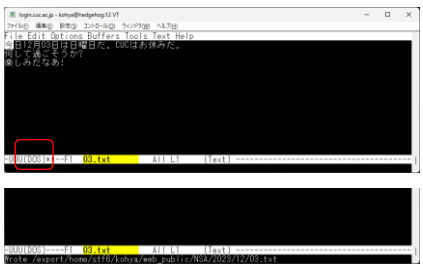
- と言ってくるので、ここで「ファイルを保存する文字コード」を選ぼう
- 選ぶべきコードは「utf-8-dos」



40

すると

- ここが変わる
- この状態でC-x C-sでファイルを上書き保存



41

で、今度は

- 03.txtをWinSCPを使い、Windowsに持ってきてメモ帳で開く
- すると今度は改行コードが「CRLF」になる

42

先週のお題

- 全部のファイルをひたすらmoreで見ていくのはしんどい
- ならば、どうすれば「あたり」のファイルを探せる？
- 「ハズレのファイル」は、「ハズレ」とだけ書かれている
- では当たりのファイルは？

43

ls –lRで探す

- 全部のファイルの「ファイルサイズ」をみてやって特徴を調べる？
- なんかもう少し賢い方法はないのか？

- 「あたりのファイルには、数字だけが書き込まれている」

44

コマンド：grep

- ファイルの中の特定の文字だけを抜き出すコマンド
- 先ほど作った20.txtの中から、「H」が含まれる行だけを抜き出す場合
  grep H 20.txt
  とする

45

とすると

- あの200個以上のファイルの中から、特定の数字(自分の学籍番号末尾一桁)が含まれたファイルを抜き出すこともできるのでは？
- grep 2 *.txt
- とすると、特定のディレクトリ以下で2という数字が含まれているファイルだけを抜き出せる

46

「*」

- アスタリスク　と読む
- UNIX上で、全てのファイルを指し示す方法
- grep 4 *.txt */*.txt */*/*.txt */*/*/*.txt
  などとして見る
- これで、現在のディレクトリから4階層下までのすべての.txtファイルを検索できる

- えーでももう少し賢い方法はない?

47

findコマンド

- findには、いろんな「条件」を足して検索する方法がある
- ls –lRを使って目で探してもいいけど…
- find . –print
  で、.（カレントディレクトリ）以下のすべてのディレクトリとファイルを表示することができた
- find . –type f
  で、「ファイルタイプが『ファイル』のみ」を検索して表示できる

48

findコマンド応用

- find . -size -10c
  で、ファイルサイズが10 character(10文字)以下の
  ファイルのみを抽出する
- find . -size -10c -exec コマンド名 {} ¥;
  で、10文字以下のファイルのリストを、コマンドに与えることができる
- find . -size -10c -exec ls -l {} ¥;
  とすると、10文字以下、という条件に合致したファイルのみを ls –l コマンドに喰わせることができる

49

では

- find . -size -10c -exec ls {} ¥; -exec cat {} ¥;
  としたら出てくるものはなんだろうか？
- ※複数のexecオプションを同時に取ることができる
- あるいは
  find . -size -10c -exec grep 4 {} ¥; -print
  としたら出てくるものはなんだろうか？

50

UNIXコマンドでも

- Excel関数でもなんでもそうなんだけれど
- **「楽をする方法(コマンド)があるはずだ」**
- **「自分が今辛いなら、誰かが既に同じ辛さを味わっているはずで、その人はもしかしたら解決策を見つけていたかもしれない」**
- と考えることはとても大事

51

ポイント

- ×**「以前に楽な方法で答えを見つけた人の答えをパクろう」**
- ○**「以前に楽な方法で答えを見つけた人の手法をパクろう」**

- 似てるようで違う
- システムエンジニアリング業界では、後者はむしろ積極的に推奨される
- とにかく「ググれ」

52

実は

- grepには –rオプションがある
- サブディレクトリ以下を再帰的に検索する
  (ディレクトリの中のファイルを検索するときに、ディレクトリが出てきたらさらにその中も検索する…といった具合)
- ls –R や、cp –rオプションと似てる
- Recursive の略

53

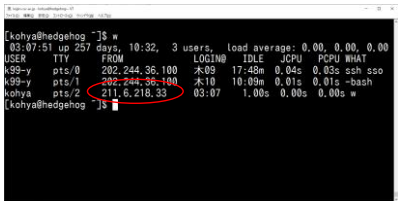
再帰の弱点

- あんまりディレクトリ構造とファイル数が多いと、探す箇所が多すぎるため、極端な速度低下を引き起こす
- が、今回に限って言えば
- grep –r 4 *.txt
  とすると多分一番早い

54

コマンド:w

- 誰が何をしているか、を調べるコマンド
- と同時に、「何処からログインしているか」を調べるコマンドでもある



55

IPアドレスは

- IPネットワークのEnd to End (始点と終点)で通信を成立させる仕組み
- 従って、wコマンドで出てくる「何処からログインしているか」を示すIPアドレスは、自分の端末のIPアドレスのはず
- しかし、実際にはNATの所為で、始点(自分のIPアドレス)と必ずしも同一とは限らない

56

コマンド: zip , unzip

- ~kohya/NSA06.zipというファイルがある
- それを自分のホームディレクトリにコピーしてください
  - やりかたはわかるよね？
- で、
  $ unzip NSA06.zip

  とすると、zipファイル(圧縮ファイルが解凍できる)
- 圧縮は
  $ zip –r 圧縮ファイル.zip 圧縮ディレクトリ

57

コマンド:history

- コマンドの履歴を一覧表示する
- どういう順番でどのコマンドを打ったか、を確認したい場合historyコマンドを使うとよろし

58

本日のお題(以下の手順を理解し実行する)

- login.cuc.ac.jpのweb_public/NSA/2023/12/02.txtをemacsで開く
- 02.txt(現在は文字コードUTF-8/改行コードLF)に対してファイルの中身の文言はまったく同じもので…

59

以下のファイルを作る

- 03-s.txt(文字コードShift_JIS/改行コードLF)
- 03-j.txt(文字コードISO-2022-JP/改行コードLF)
- 03-e.txt(文字コードEUC-JP/改行コードLF)
- 03_crlf.txt(文字コードUTF-8/改行コードCRLF)
- 03_cr.txt(文字コードUTF-8/改行コードCR)
- 03-16be.txt(文字コードUTF-16-BE/改行コードLF)
- 03-16le.txt(文字コードUTF-16-LE/改行コードLF)

60

このファイルサイズについて

- 自身で比較をし、考察を書きなさい
- https://forms.gle/Cd99jgrDv6yg7pgbA

61