TSUBAME利用法 入門編(Linux基礎)

目次

- TSUBAME4概要
- Linuxの概要
- ファイル操作
- 各種コマンド
- プログラムの実行
- 利用環境

TSUBAME4概要

- TSUBAME4とは
- 機器詳細
- 商用アプリケーション
- サービス内容
- 計算ノードの利用
- センターホームページの活用

TSUBAME4とは

- 2024年4月1日に東京工業大学学術国際情報センター(GSIC)に導入されたスーパーコンピュータ

- 倍精度総理論演算性能:66.8PFLOPS
- 半精度総理論演算性能:952PFLOPS
- 総主記憶容量:180TiB
- 総SSD容量:327TB
- 総HDD容量:44.2PB
- 200Gbps高速ネットワーク(システム内)

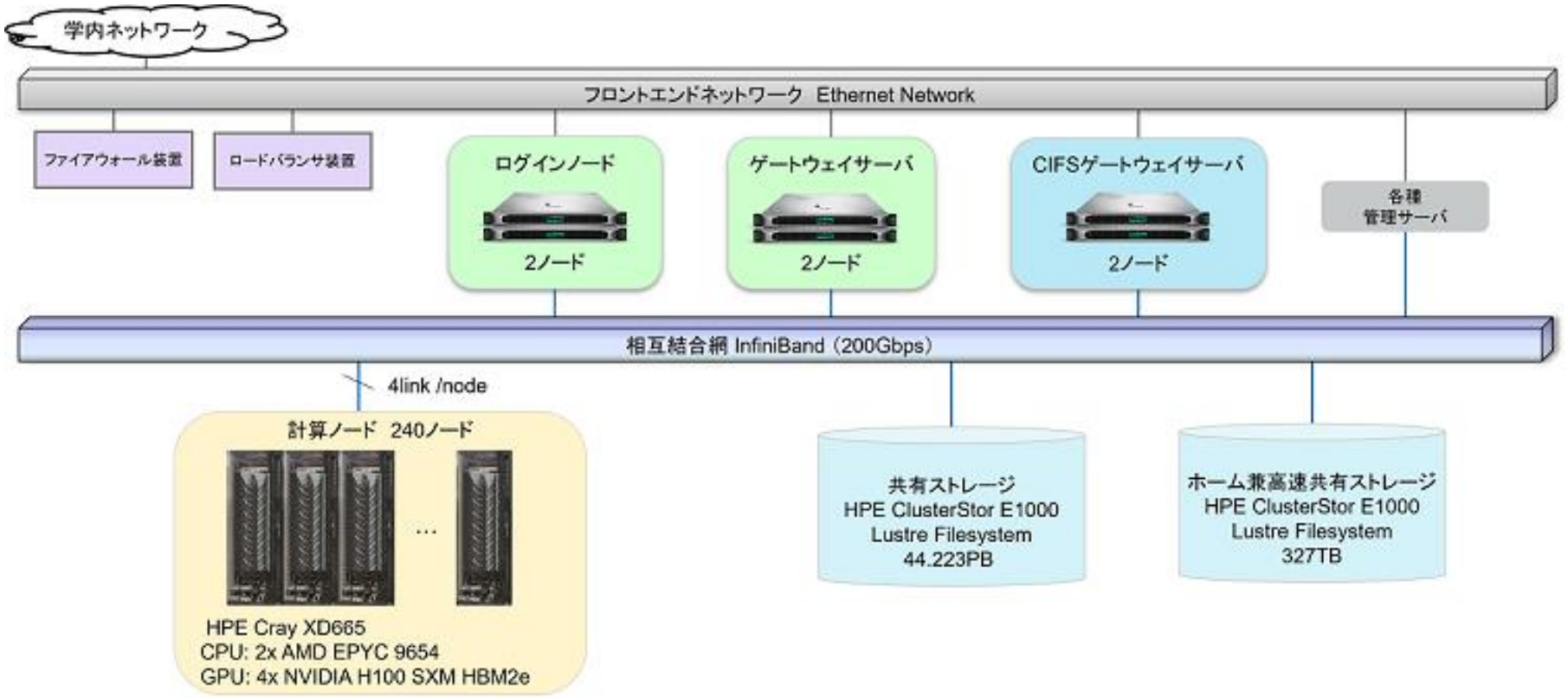
特徴

- AI時代に即した演算性能とストレージ
  - 計66.8PFlops(倍精度)の演算性能 ・・・ TSUBAME3.0の5.5倍
    NVIDIA H100 GPU 総計960枚
    AI分野で注目されている、半精度なら952PFlops
    計44PBの共有ストレージ容量
    高速共有ストレージ(SSD)の追加

- 様々な利用形態への対応
  - Webベースでも、これまで通りSSHでも
  - 4コアの資源タイプから高並列・マルチGPUジョブ向けまで

- TSUBAME3などのソフトウェアはほぼそのまま稼働
  - x86_64 CPU + NVIDIA GPU + Linux OSという組み合わせ

TSUBAME4システム概念図



機器詳細（1）

- 計算ノードHPE Cray XD665 240ノード

演算部名		計算ノード　240台
ノード構成（１台あたり）	CPU	AMD EPYC 9654 (2.4GHz) × 2CPU
	コア数/スレッド	96コア / 192スレッド × 2CPU
	メモリ	768GiB
	GPU	NVIDIA H100 SXM5 94GB HBM2e × 4
	SSD	1.92TB NVMe U.2 SSD
	インターコネクト	InfiniBand NDR200 200Gbps × 4

機器詳細 (2)

- ストレージ

用途	マウント	容量	ファイルシステム
高速ストレージ領域 Homeディレクトリ	/gs/fs /home	372TB	Lustre
大容量ストレージ領域 共有アプリケーション配備	/gs/bs /apps	44.2PB	Lustre
ローカルスクラッチ領域	/local	各ノード1.92TB	xfs（SSD)

- ソフトウェア
  - OS:RedHat Enterprise Linux Server 9.3
  - 商用アプリケーション/別ページ参照

商用アプリケーション一覧

ソフトウェア名	概要
ANSYS	解析ソフトウェア
ANSYS/Fluent	解析ソフトウェア
ANSYS/LS-DYNA	解析ソフトウェア
ABAQUS	解析ソフトウェア
ABACUS CAE	解析ソフトウェア
Gaussian	量子化学計算プログラム
GaussView	量子化学計算プログラム プリポストツール
AMBER	分子動力学計算プログラム
Materials Studio	化学シミュレーションソフトウェア(材料系)
Mathematica	数式処理ソフトウェア
COMSOL	解析ソフトウェア
Schrodinger	化学シミュレーションソフトウェア
MATLAB	数値計算ソフトウェア
Arm Forge	デバッガ
Intel oneAPI Compiler	コンパイラ
NVIDIA HPC SDK Compiler	コンパイラ

提供アプリケーションはmoduleコマンドで環境のロード/パージが可能

Gaussian,デバッガおよびコンパイラ**以外**は原則学内ユーザのみ利用可能

TSUBAME4.0のアプリケーション

- 商用アプリケーションの利用に対するポイント課金あり
- TSUBAMEポータルから各ユーザごとに申請が必要
- ユーザの申請権限はグループ管理者が付与する

サービス内容

- 計算ノード利用
  - 240ノードの計算ノードHPE Cray XD665を利用可能
    計算ノード利用イメージは，別ページ参照

- ストレージ利用
  - homeディレクトリ(制限 25GiB)：無償
    - 各自のホームディレクトリは，/home/［0-9］/アカウント名
    - システムの全マシンから，アクセス可能
  - 高速/大容量ストレージ領域(購入期間中は永続)：有償
    - Lustreファイルシステムで構成されるグループディスク領域
    - 高速はSSD,大容量はHDDで構成されている
  - スクラッチ領域(ジョブ利用中のみ)
    - ローカル/共有スクラッチ領域
      - SSDで構成されるローカルスクラッチ領域、ジョブが終わると消滅

有償サービスの範囲

- 計算ノードを使った計算(従量制の利用)
- 計算ノードの事前予約
- 定額制による月単位のノード確保(TSUBAME4で導入)
- 一部商用アプリケーションの利用(TSUBAME4で導入)
  - TSUBAME4内での利用 (月額制・ユーザ毎)
  - アプリケーション配布を受けて研究室内で利用 (年額制・研究室毎)
- TSUBAMEの有償サービスを利用する場合、「TSUBAME4 ポイント」というものをTSUBAME4ポータルで購入して、上記各サービスと交換

TSUBAME4ポイントの有効期限は年度内・繰り越し不可

TSUBAME利用の流れ

[受講者]は2まで終了済みという前提
円滑な利用のためにはTSUBAMEポイントが必要となります。

1. TSUBAMEアカウント取得
2. SSH鍵ペアの作成と公開鍵の登録(本講習はパスワード利用)
3. グループの作成[グループ管理者のみ]
4. グループの各種設定
   - 支払いコードの登録　[グループ管理者のみ]
   - ポイント購入　[グループ管理者(サブを含む) 、権限付与された物]
   - グループへのユーザを追加　[グループ管理者(サブを含む) 、権限付与された者]
   - グループメンバへの権限付与　[グループ管理者(サブを含む)]
   - グループディスクの設定　[グループ管理者、権限付与された者]
   - 予約設定　[グループ管理者(サブを含む) 、権限付与された者]
   - アプリケーションの利用申請　[グループ管理者、権限付与された者]

   …上記は一例
5. 利用(ジョブ投入、OOD利用)

学内は基本的に
TSUBAMEポータルで実施

計算ノードの利用について

- 大きく下記の2種類あります。
- 本講習はLinux講習のため前者の利用について主に紹介します。
  - SSH接続をつかったCLIでの利用
  - Open onDemandを使ったWebブラウザ上での利用

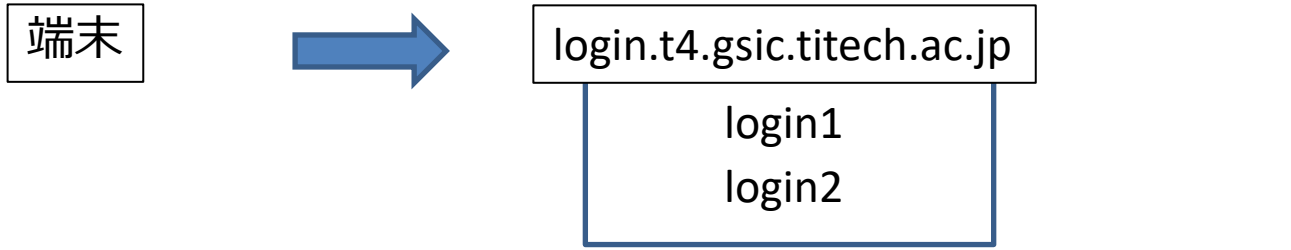
計算ノード利用(SSH)

- ログイン(SSH鍵認証のみ)
  - 研究室内PCなどより，sshコマンドを実行.
    自動的にログインノードのいずれか(login1/login2)にログインされる.

※ログインノードでは計算、サーバ起動など負荷がかかる利用は禁止されており、制限にかかった場合は自動的に実行しているプログラムが停止されます。VS Code等サーバを起動するような高負荷なIDE環境を利用している場合はログイン時に多くのプロセスを生成しログイン後に何もできない状態になったりします。

※ログインに使うSSH公開鍵をアップロードする必要がある

「TSUBAME4.0ポータル利用説明書」参照



負荷分散によりloginノードのどちらかにログイン

計算ノード利用(SSH)

ジョブは240ノードある計算ノードのいずれかで実行される

ユーザによるノード指定は基本的に出来ない

- インタラクティブジョブ(小、中規模ジョブ)
  - ログインノードから，ジョブ投入用コマンドqrshを利用して実行

    $ qrsh -l資源タイプ-l 確保時間 -g TSUBAME4グループ
  - インタラクティブノード上で実行

    $ ./a.out
- バッチジョブ(中、大規模ジョブ)
  - ログインノードから，ジョブ投入用コマンドqsubを利用して実行

    $ qsub -l 資源タイプ -l確保時間 -g TSUBAME4グループ job.sh

ノードは，いくつかの資源タイプに分かれている

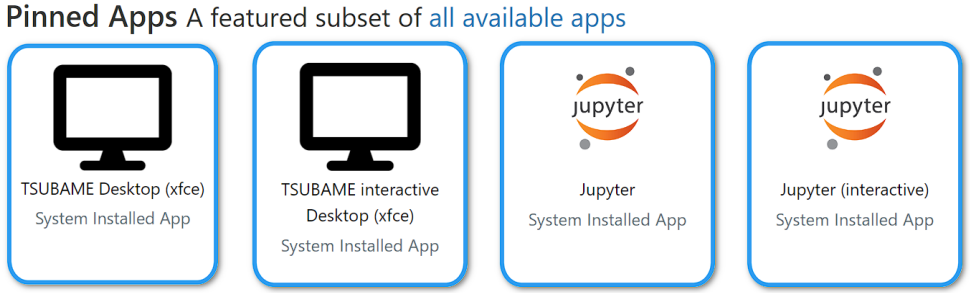
→資源タイプは，「利用の手引き」参照

計算ノード利用(Open onDemand)

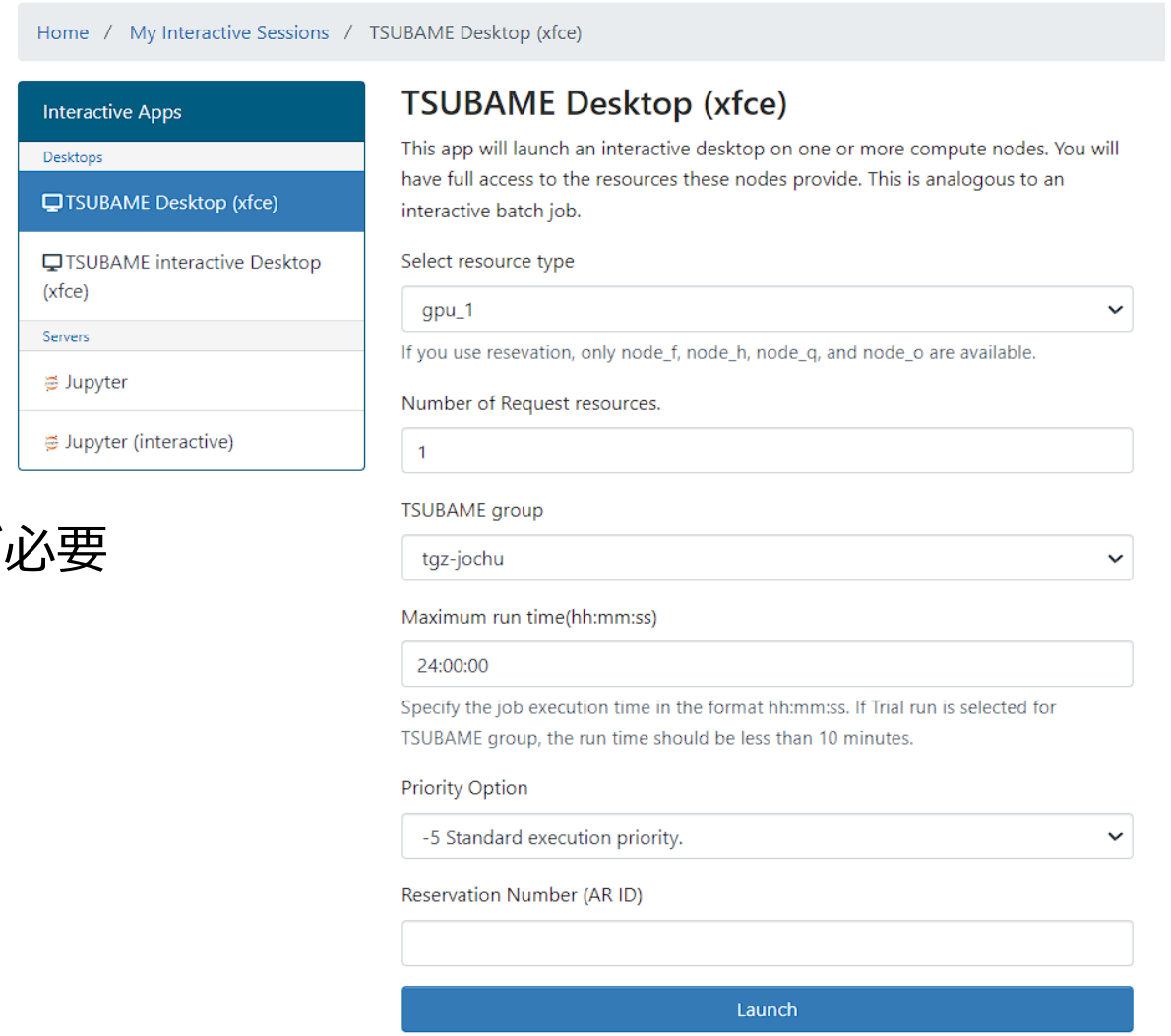
- https://ood.t4.gsic.titech.ac.jp/

intaractiveとついているappは学内無償
(インタラクティブキューの制限)

利用にあたってはパスワードの登録と
TSUBAMEに登録したメールが受け取れる環境が必要



詳細はOpen OnDemandの利用の手引きを参照
https://www.t4.gsic.titech.ac.jp/docs/ood/



Webページなどのご紹介

- TSUBAME4計算サービス
  - https://www.t4.gsic.titech.ac.jp/
    全ての案内はここから到達可能
- TSUABME4ポータル
  - https://portal.t4.gsic.titech.ac.jp/
    SSH鍵登録・TSUBAME4ポイントの購入など
- X (旧Twitter)
  - @titech_Tsubame
    速報的な情報の広報は計算サービスサイトのトップページとXにて行っております。
    問い合わせはTSUBAME4計算サービスページ内、「お問い合わせ」のウェブフォームまで

過去の回答メールのアドレスに直接送信はご遠慮ください

Linuxの概要

- Linuxの概略
- ログイン
- リモートログイン、ログアウトの仕組み
- ターミナル
- TSUBAMEへのログイン
- ログアウト

Linuxの概略

- LinuxはLinuxカーネルを利用したOS

- Linuxはマルチタスク/マルチユーザーのOS
  - 複数のユーザーが同時に使える
  - 利用者が誰か、正当な利用者かどうか

- 多様性の高いLinuxディストリビューション
  - debian
  - slackware linux
  - SUSE Linux Enterprise Server (SLES)
  - Red Hat Enterprise Linux (RHEL)
  - CentOS
  - …

ログイン

- 利用者が誰か、利用者が正当な利用者であるかどうかの認識をする。
- Linuxにログインするときはログイン名とパスワードが必要となる。

_____-000 へようこそ
ログイン名 : _____
パスワード : _____

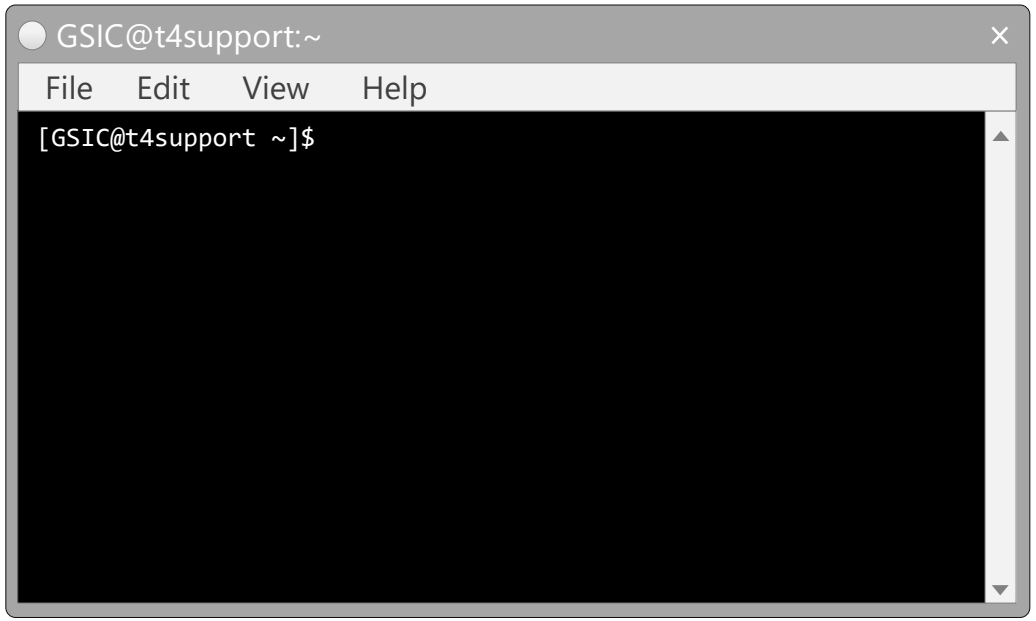
必要事項を入力後に認証されて使用可能となる

リモートログイン

- 遠隔地にあるホストを、現在のホストから使用する
- コマンドとしては telnet，rlogin, ssh など
- TSUBAMEではセキュリティの高い鍵認証sshを導入

端末(ターミナル)

- Linux利用時は標準的な端末(ターミナル)を使用します
- 古くは汎用機などへ接続するためのハードウェアである端末装置のことを指したが、ソフトウェア化した仮想端末のこと
- ターミナルはCLI（コマンドラインインタフェース ）のための環境を提供し、CLIでは処理の自動化が容易でネットワーク負荷も低く、自動補完・履歴確認、入出力の切り替えの容易さや先行入力が可能といった、なれたユーザにとっては多くの利点があります。



Windows用ターミナルからの利用

- Cygwin/PuTTY/RLogin/MobaXterm/WSL…など様々なソフトウェアがあります。

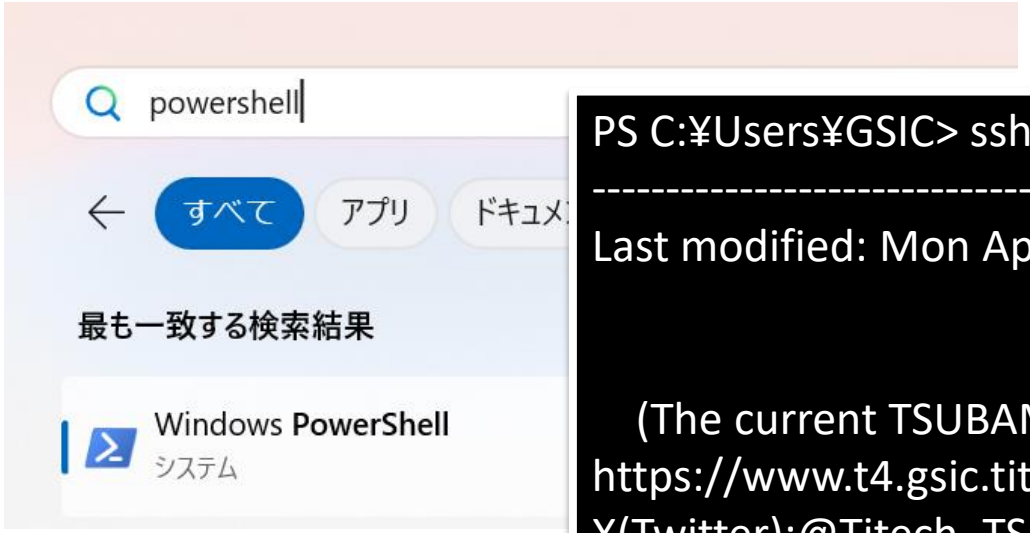
- おすすめはRLogin/MobaXtermですがいろいろ試して気に入ったものをお使い下さい。

各ターミナルアプリケーションの利用方法についてはお問い合わせいただいてもお問い合わせ窓口ではサポートできません。
個人環境となるため、出来ても助言程度となります。

参考:Windowsで利用できるSSHクライアントについて

Windows Powershellでの接続例

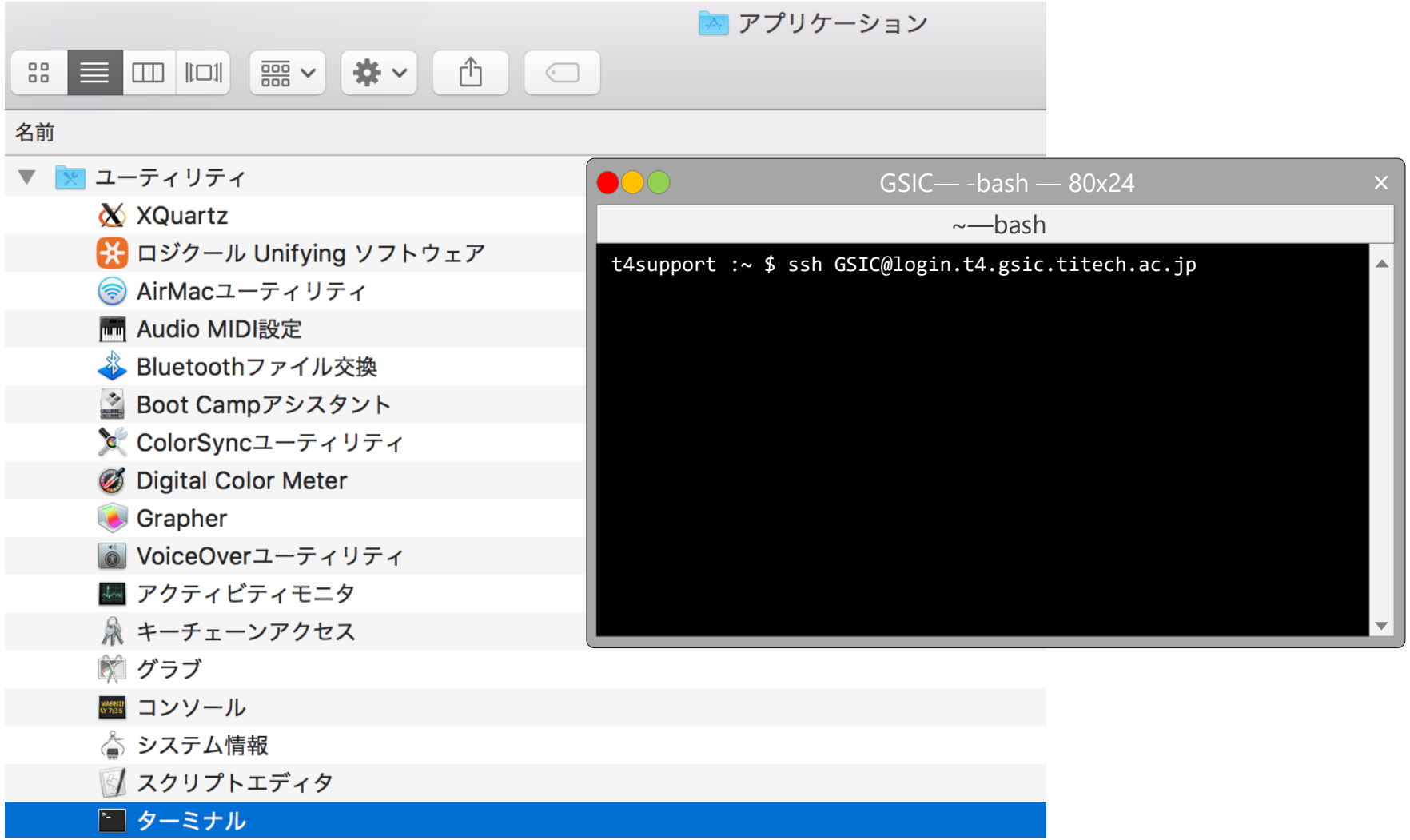
- スタート＞検索窓にpowershellと入力しEnterを叩いてpowershellを起動する
- SSHコマンドを実行する。(鍵ペアを作成していない場合は作成を行う)




```
PS C:¥Users¥GSIC> ssh ux00000@login.t4.gsic.titech.ac.jp
---------------------------------------------------------------
Last modified: Mon Apr  1 10:00:09 JST 2024


    (The current TSUBAME 4.0 operational status)
https://www.t4.gsic.titech.ac.jp/
X(Twitter):@Titech_TSUBAME
---------------------------------------------------------------

Last login: Tue Apr 23 10:55:08 2024 from 10.29.2.11
[ux00000@login1 ~]$
```

Macからの利用

- アプリケーション>ユーティリティ>ターミナル.app



SSH鍵認証

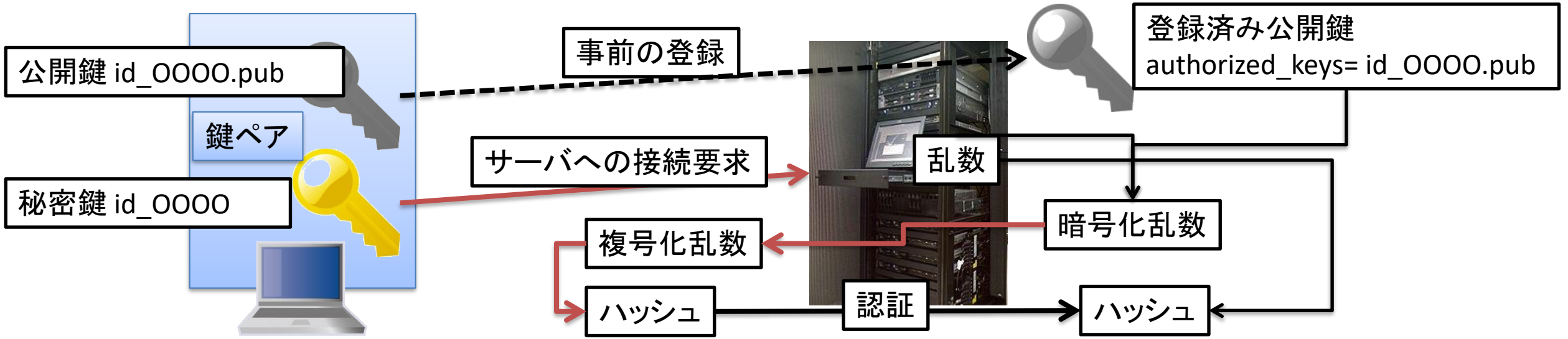
公開鍵(Public)/秘密鍵(Private)のペアを利用した認証システム

公開鍵はid_{暗号化方式}.pub,秘密鍵はid_{暗号化方式}といったファイル名

暗号化方式は

- メリット
  - 鍵を所有しているマシンからのみアクセスが可能=高セキュリティ
- デメリット
  - 作成/管理のコスト

*秘密鍵は絶対に他人に公開しないようにしましょう。



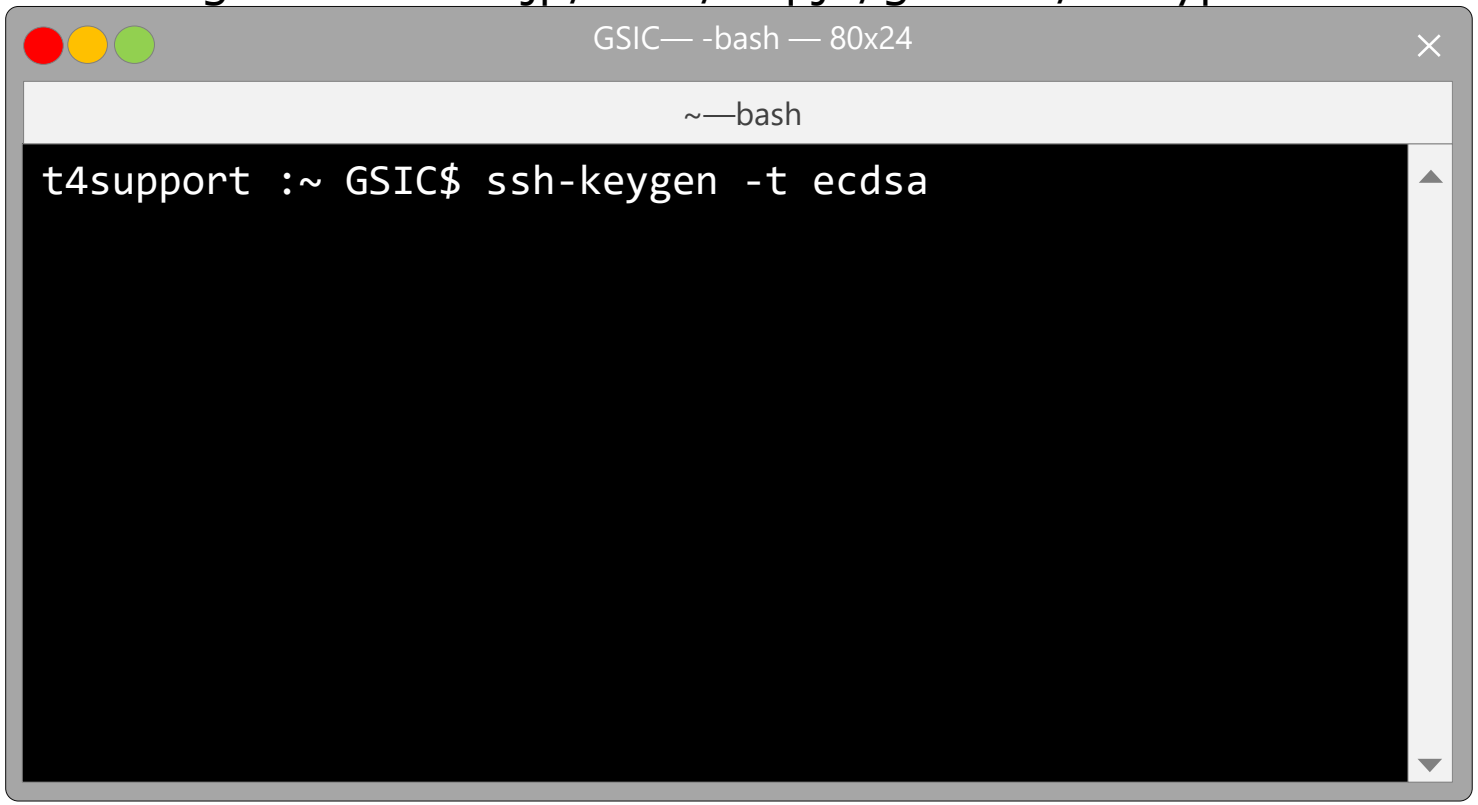
鍵の作成/登録

- 鍵を作成しましょう(実習室では実施しないで下さい)

  まず、端末を起動してください。

  起動していたらターミナルを起動して以下のコマンドを打ってください。

  参考：https://www.t4.gsic.titech.ac.jp/docs/faq.ja/general/#keypair



鍵の登録

ポータルに登録するので、以下のサイトにアクセスします。

- https://portal.t4.gsic.titech.ac.jp/ptl/user/sshPublicKey

以下のコマンドをターミナルで打ち、表示された内容をコピーし、追加ボタンをクリックします。

TSUBAME4へのログイン

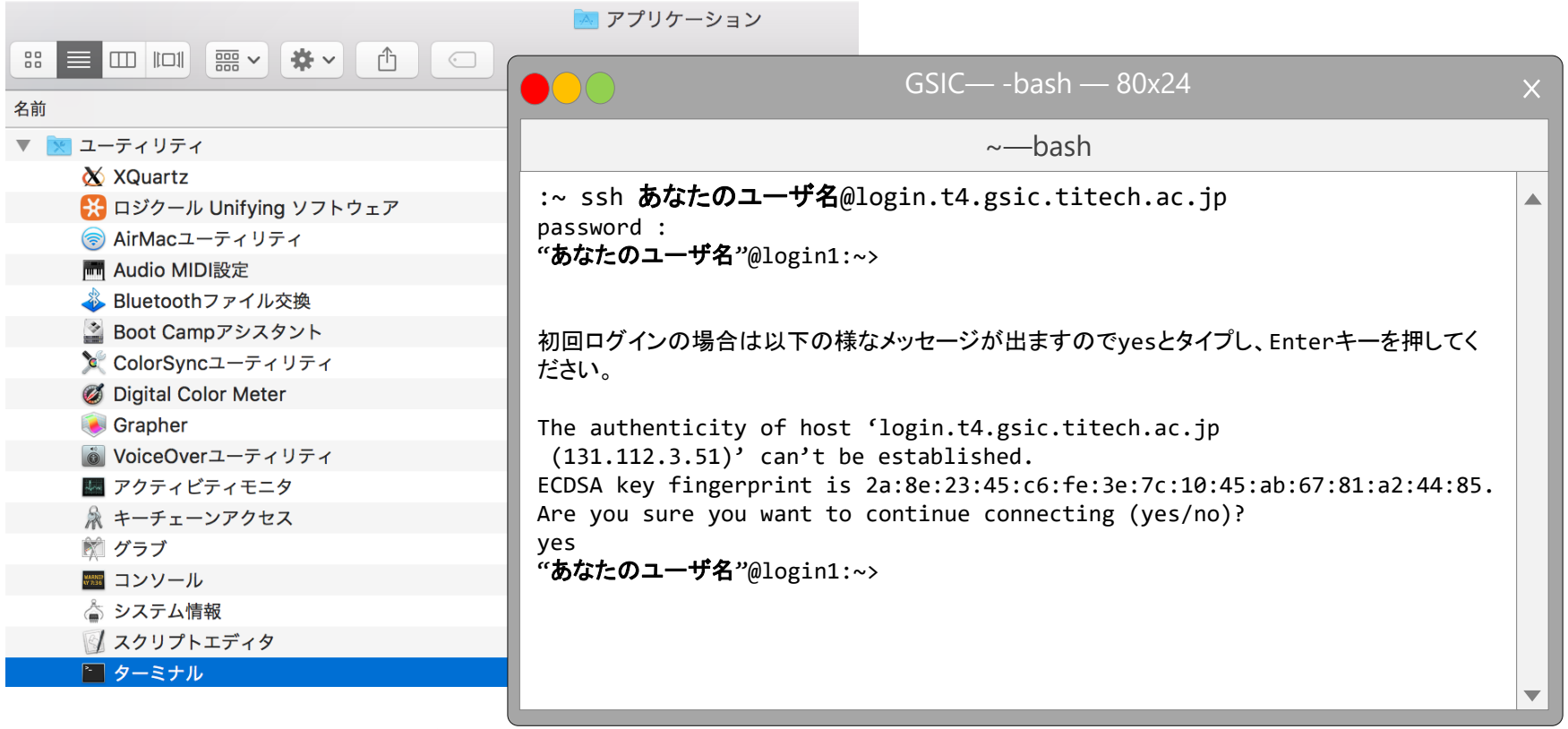
- ログイン操作 → % や $ という記号を表示(ターミナル)
- これをプロンプトと呼び、コマンドを入力できる状態
- この状態からTSUBAMEを使うためにリモートログイン

→ssh を使用

実習室ではTSUBAMEポータルに登録した「パスワード」を利用して下さい。

実際にログイン操作を行います

- アプリケーション>ユーティリティ>ターミナル.app



実習室ではパスワードログインとなりますので、ポータルに登録した「**パスワード**」を利用して下さい

ユーザはux00000のような

TSUBAMEログインノードのlogin1もしくはlogin2にログインできていればOKです

ログアウト

- 操作を終了したい場合はログアウト操作
- 現在作業しているマシンの確認
  - データの整理
  - 実行状態の確認

- ログアウト操作
  - Ctrlキーと小文字のdを同時に入力(Ctrl-dと表現)
  - logoutコマンドを使用
  - exitコマンドを使用

Tips

- 研究室や自宅から自分が利用するクライアントから接続する際は、KeepAliveなどの設定を適切に行なって下さい。

- 行わない場合はTSUBAMEとの通信が切断されます。

- クライアントによって作業が異なります。
  - mac,cygwin,linuxでは以下のような設定を追記する

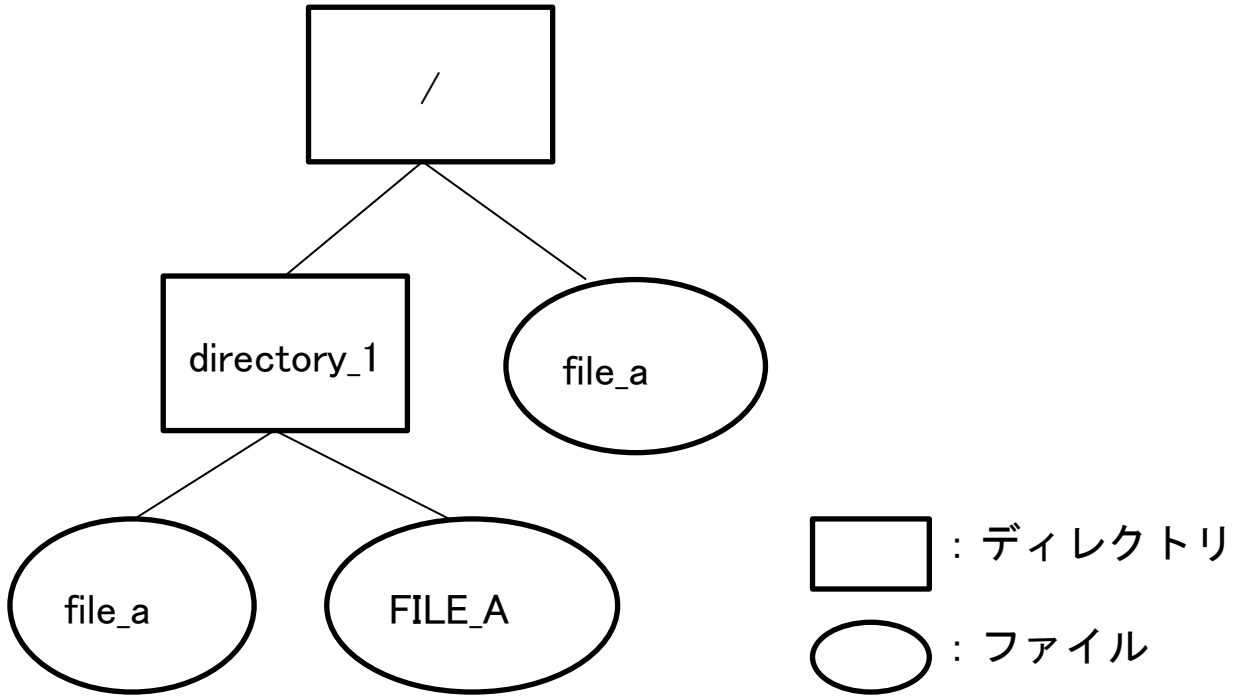
  - sshコマンドに-o ServerAliveInterval=120をつけても良い

ファイル操作

- ファイルシステム
- ファイルの指定
- ファイル指定に使用する記号
- 情報表示コマンド
- ファイルを指定する特殊文字
- ファイルのパーミッション
- ファイル管理コマンド
- マシン(OS)によるファイルの差異

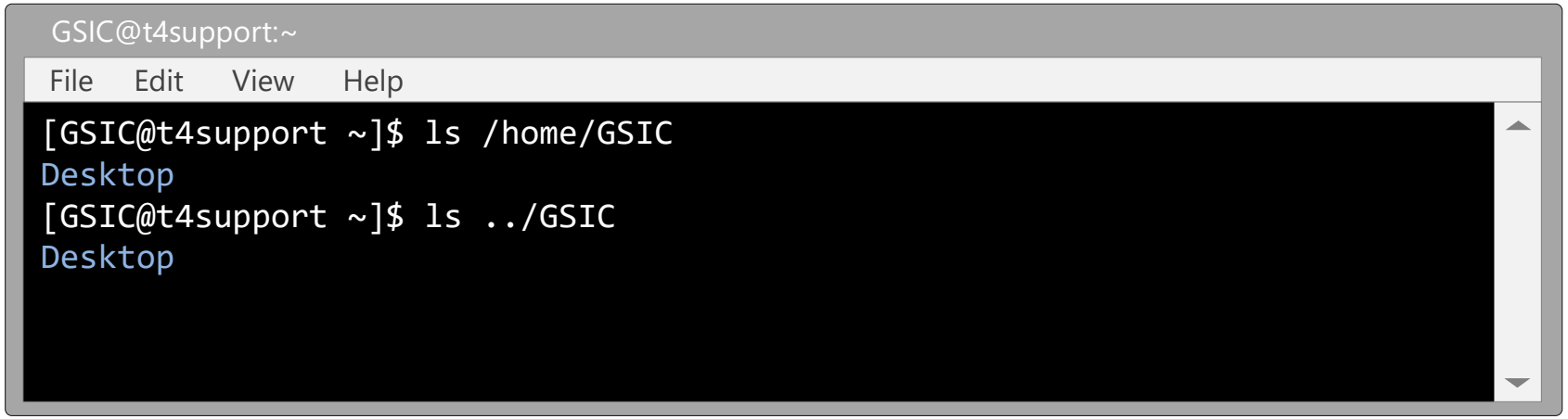
ファイルシステム

- 階層構造で表現
- ディレクトリによるファイル管理
- 一般ファイルと特殊ファイル



ファイルの指定

- 階層構造のトップを / と表現し、”ルート”と呼ぶ
- ファイルへのアクセス方法
  - 絶対指定：ルート(/)からフルパス指定
  - 相対指定：自分の位置(ディレクトリ)からファイルが存在する位置を指定
    - ../　一つ上の階層を表現
    - ./　現在の階層を表現



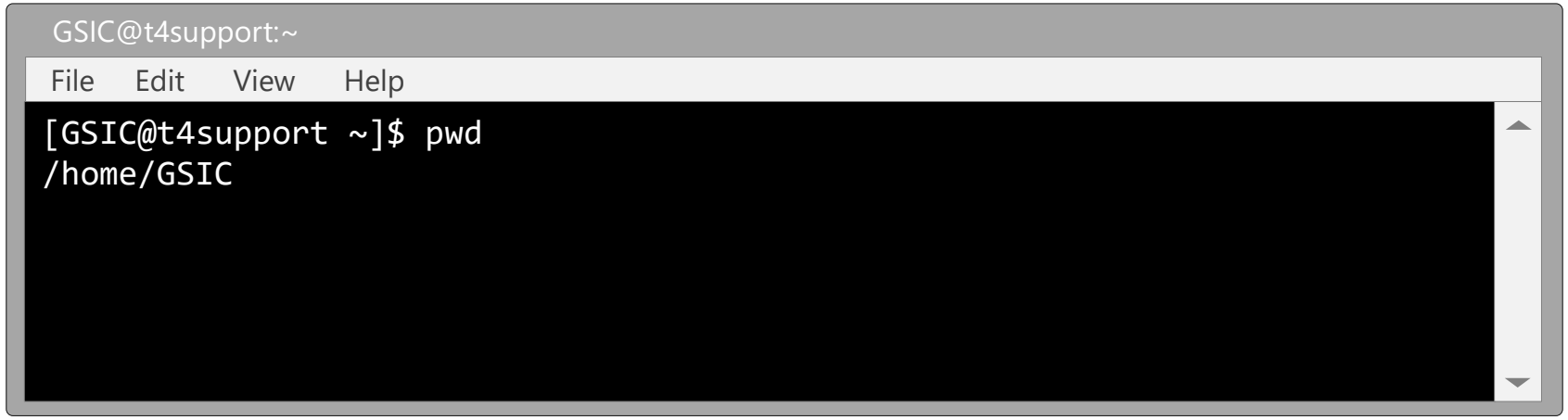
ファイル指定に使用する記号

- ホームディレクトリ　　~
- 一つ上のディレクトリ　../
- 現在のディレクトリ　　./

※カレントディレクトリ(現在位置)を表現する場合．(ドット)を使用

情報表示コマンド

- pwd (自分が今何処にいるのかを表示するコマンド)



- ls　(欲しい情報に合わせてオプションを指定)

ファイルを指定する特殊文字

- ファイル名を指定するために特殊文字表現が利用可能。

メタキャラクタ	機能	使用例
*	全ての文字列に対応	ls *
?	1文字に対応	ls ?
【文字列】	文字列の中の1文字に対応	ls bc
【文字1-文字2】	文字1と2の間にある1文字に対応	ls b[a-c]d

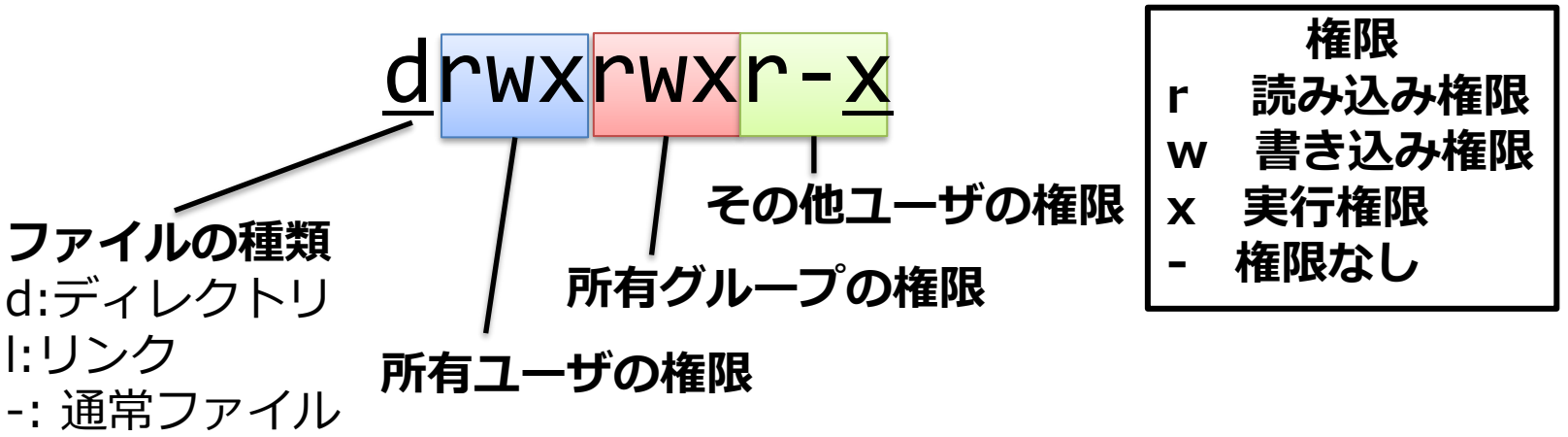
※メタキャラクタ
それ自体は意味を持たないが、他の文字と組み合わせることで全体で
意味を持たせる働きを持つ記号のこと。メタ文字とも呼ぶ。

ファイルのパーミッション

- ls –l コマンド(-lオプション付き)で確認

```
GSIC@g3support2:~
File  Edit  View  Help
[GSIC@t4support hogehoge]$ ls -l
合計 0
drwxrwxr-x. 2 GSIC users 6 10月  3 15:35 hoge
-rw-rw-r--. 1 GSIC users 0 10月  3 15:35 hogedoc
モード  サイズ 所有者 グループ 日付      名称
```

- モードについて



ファイル管理コマンド

- ディレクトリの作成　　% mkdir aaa
- ディレクトリの削除　　% rmdir aaa
- ファイル属性の変更　　% chmod 755 aaa

※ 755はオクテットまたはビット表現と呼び、所有者、グループ、その他のユーザーに対するアクセス権を表現する
0　– (権限なし)
1　x (実行可)
2　w (書き込み可)
3　1+2
4　r (読み込み可)
5　1+4
6　2+4
7　1+2+4
(以下はマシン管理者の操作)

- 所有者の変更　　% chown necapps aaa
- グループの変更　　% chgrp procon aaa

マシン(OS)によるファイルの差異

- テキストファイルの改行コード
  - Unix/linux　　¥n LF(line feed)
  - Macintosh　　¥r CR(carrage return)
  - Win/Dos　　¥r¥n CRLF

- 文字コードは、日本語の表示コード
  - Unix/Linux/macOS UTF-8/EUC
  - Windows UTF-8/Shift-JIS

- nkf コマンドにより変換(UTF-8に変換)
  nkf –w abc.txt > abc_utf8.txt

- 改行コードがWin/Dosのままだと Unix/Linuxでは動きません
  nkf コマンドによる変換(改行コード変換)
  nkf -Lu abc_*crlf.sh* > abc_*fl.sh*

圧縮・解凍

- 圧縮

  gzip atom45.tar → atom45.tar.gz
  zip atom45.zip atom45 → atom45.zip
  tar czvf atom45.tgz atom45 → atom45.tgz
  tar cjf smpl.tar.bz2 smpldir → smpl.tar.bz2
  bzip2 sample.txt → sample.txt.bz2

- 解凍

  zcat atom45.tar.Z | tar -xvf -
  tar jxf sample.tar.bz2
  gzip -d atom45.tar.gz → atom45.tar
  unzip book2nd.zip
  tar xzvf atom45.tgz
  bzip2 -d sample.txt.bz2 → sample.txt

各種コマンド

- よく使用するコマンド
- コマンド操作
- ファイル操作コマンド
- alias(別名)の機能
- エディタ
- コマンドの結合
- コマンドとジョブ

よく使用するコマンド

- ssh
- exit
- mkdir
- rmdir
- chmod
- chown
- chgrp
- nkf
- cd
- cp
- mv
- rm
- pwd
- ls
- vi もしくはemacs
- view
- tail
- cat ,more ,less
- find
- file
- grep
- diff ,sdiff
- man

コマンド操作

- コマンドのみで実行
- コマンドに引数(オプション)をつける
- コマンドに引数(ファイル)を指定する
- 複数のコマンドを組み合わせる

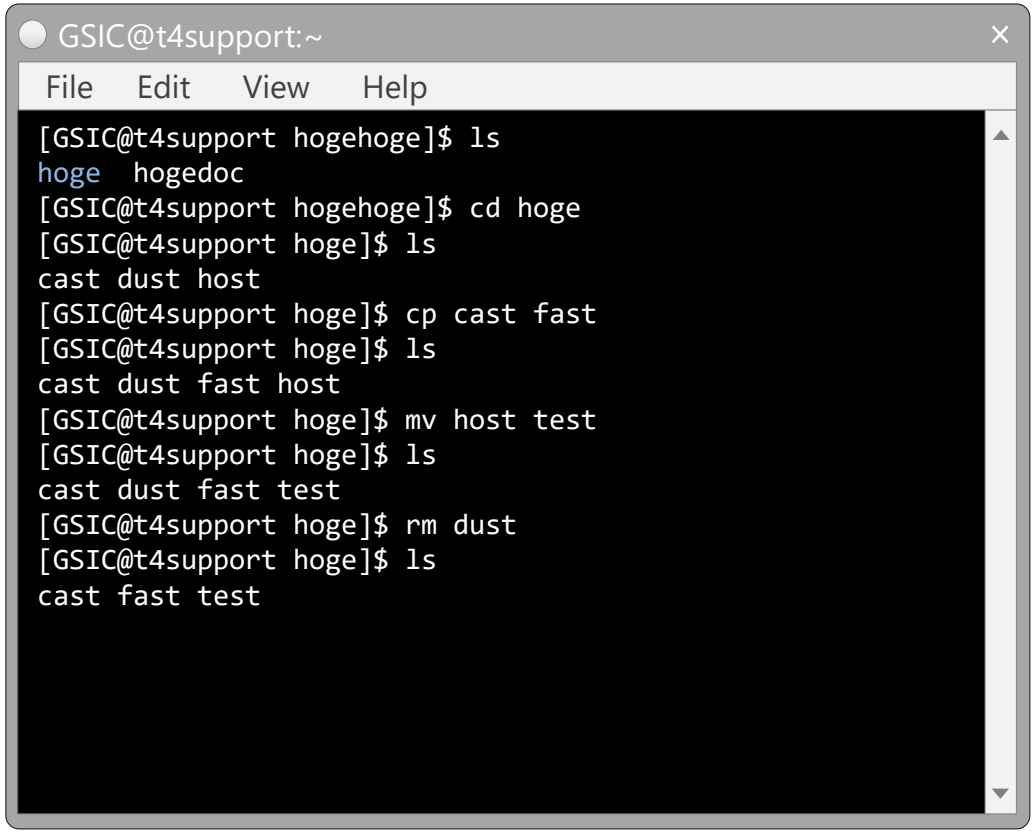
GSIC@t4support:~

File Edit View Help

```
[GSIC@t4support ~]$ ls
Desktop
[GSIC@t4support ~]$
[GSIC@t4support ~]$ ls -l
drwxr-xr-x  2 GSIC users    512 Sep 13 10:15 Desktop
[GSIC@t4support ~]$
[GSIC@t4support ~]$ cal 10 2017
    October 2017
Su Mo Tu We Th Fr Sa
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31
```

ファイル操作コマンド

- cd
  change directoryの略
  ディレクトリの移動
- cp
  copyの略
  ファイル、ディレクトリの複製
- mv
  moveの略
  ファイル、ディレクトリの移動
- rm
  removeの略
  ファイル、ディレクトリの削除



※cp,mv,rmは –i オプションで確認処理をする
※一度消去したファイルの復活コマンドはありません。

注意事項

危険なコマンド例

- **$ rm -rf ＊**

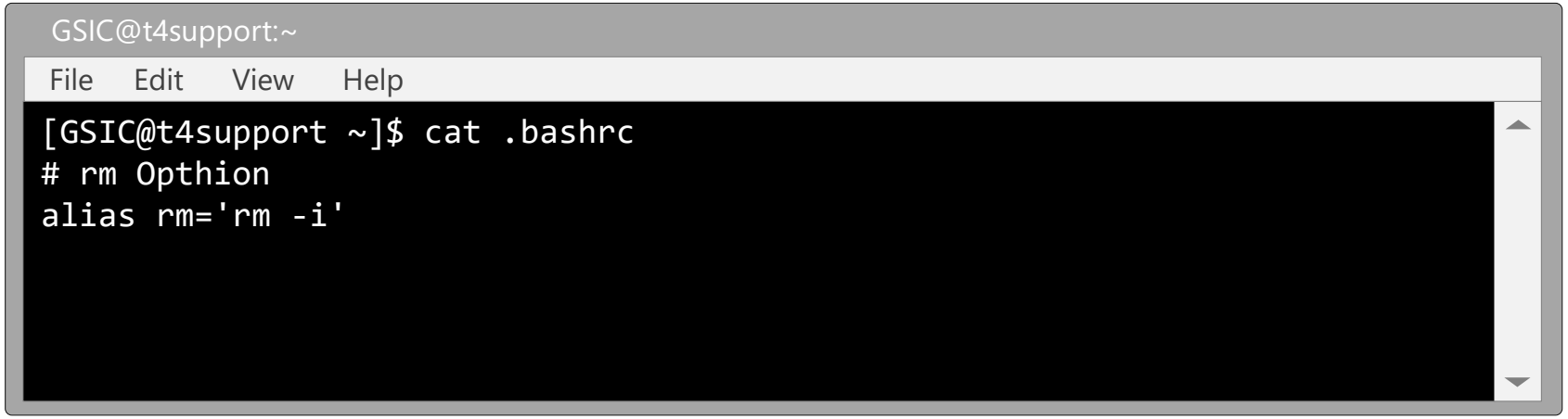
大事なものが入っているディレクトリで叩いては駄目です
＊←　全てを表す記号

スクリプト化した際には以下の記載にもご注意下さい
ホームディレクトリの中身が消え、ログインできなくなります

- **$ rm -rf ~**

alias(別名)の機能

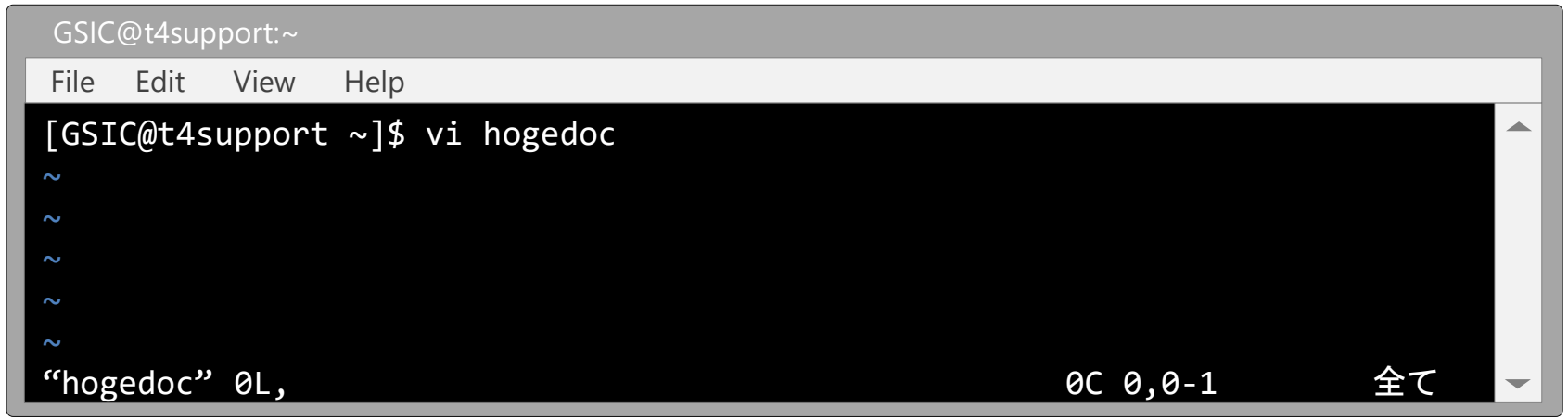
- よく使うコマンドやコマンド列に別名をつける。
- 別名もまた、コマンドとして利用できる。
- 別名を設定するコマンドを alias と呼ぶ。
- 別名定義を解除するコマンドを unalias という。



※このように.bashrcに書いておくとミスによるファイルの喪失を防止できる。
例えば、rm a* とすると、aで始まるファイルだけを削除する。
間違って、rm a * のように間にスペースが入ると、aというファイルと全てのファイルを削除してしまう。 aliasを設定することで、- i オプションが利き、削除するか確認が来る。

viエディタ

Linuxマシンにはほぼインストールされているエディタ

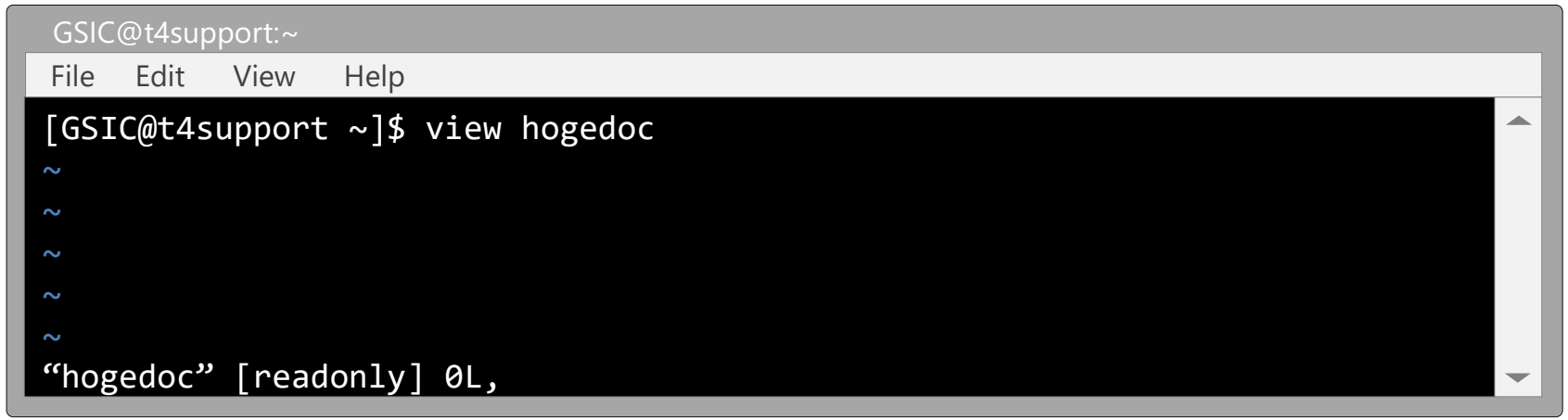


待機モード時のキー入力	効果
i または a	入力モードに変更
Esc	待機モードに変更
x	カーソル文字削除
dd	1行削除
:wq	エディタを保存終了
:q!	廃棄終了
u	一つ前に戻れる
hjkl	←↓↑→移動

待機モード時のキー入力	効果
/	文字列検索
w	ワード単位で右移動
W	スペース単位+同上
b	ワード単位で左移動
B	スペース単位+同上
Enter/Return	下の行の先頭文字に移動
r	カーソル文字を置換
yy	行コピー

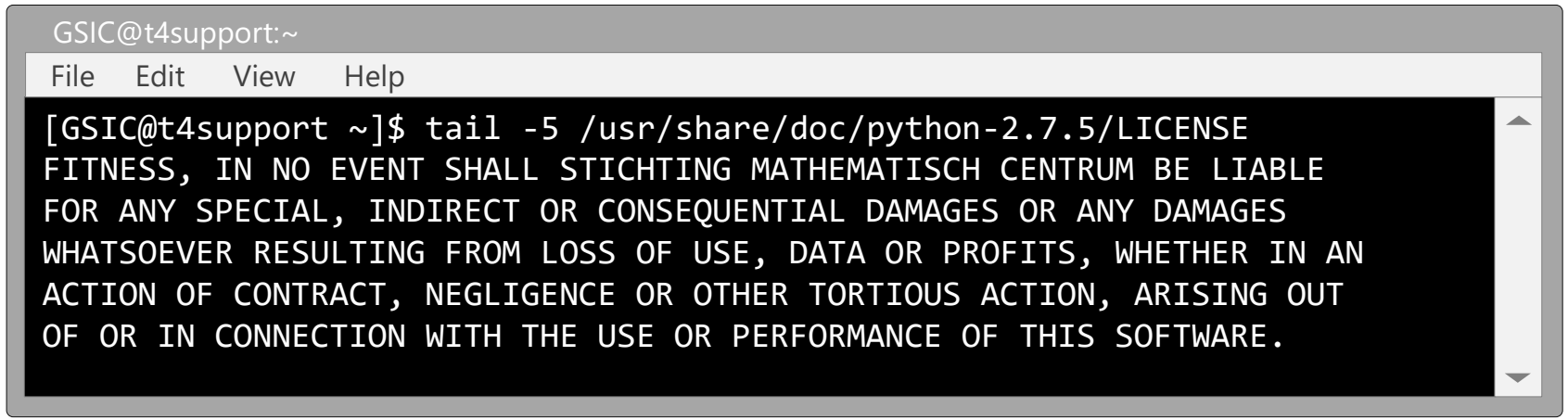
viewコマンド

- viエディタと同じエディタ内コマンドが利用可能。
- 上書き禁止モードでエディタを開く。
- ファイルの中身を確認する場合に使用。



tailコマンド

- ある出力の最後の一部分を標準出力に表示

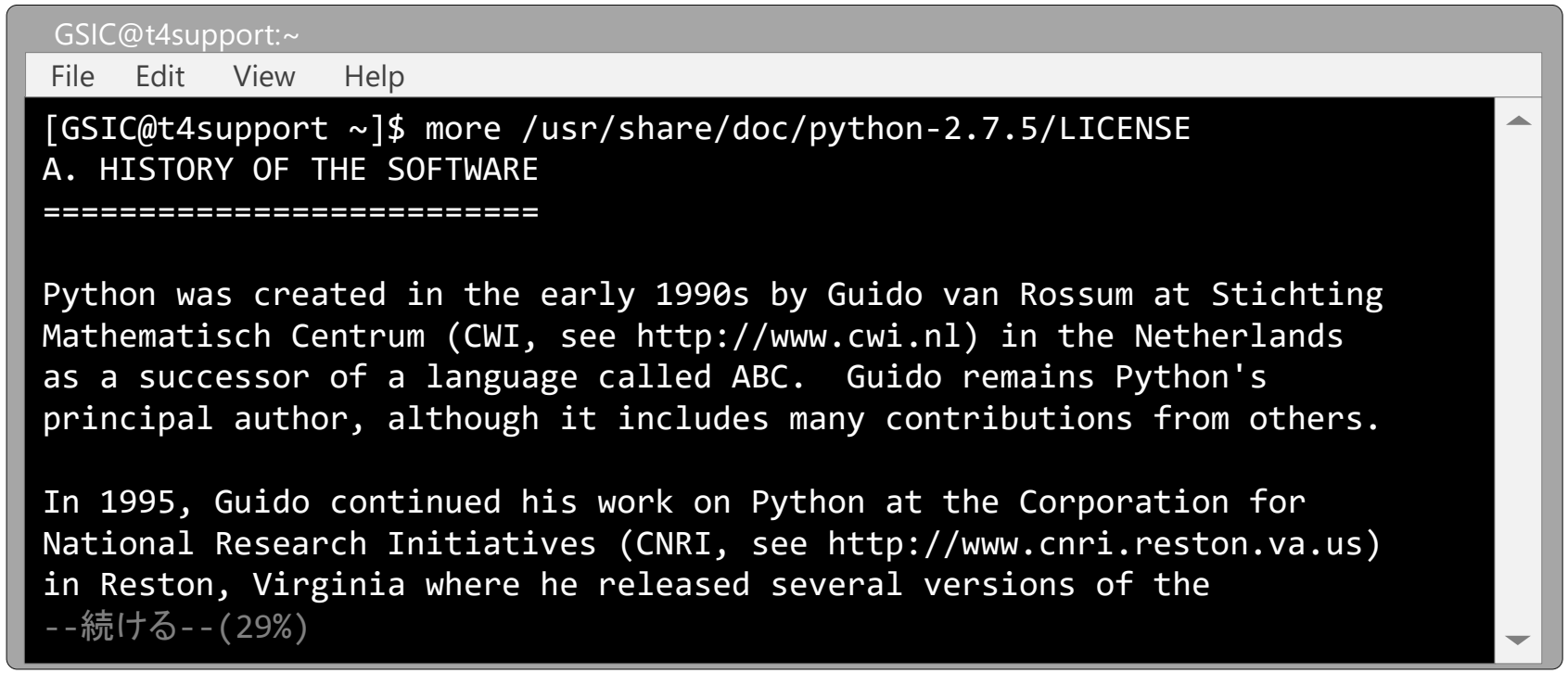


※計算結果を定期的にファイルに出力して、tailコマンドで進行状況をみる。
ただし、プログラムでのテキストの連続出力はしない
tailコマンドの連続発行はしない (マシン負荷が増大するため)

cat ,more ,lessコマンド

- ファイルの内容を表示したい場合にcatを利用
- サイズの大きいファイルをページ出力したい場合はmore,less

  more,lessはページ単位で停止するので、スペースキーでページ送りします。



findコマンド

- ファイルの存在する位置を知る
  → ファイル名やファイル名の一部がわかっている場合
  → 実行権のないディレクトリに関する検索は不可能
- 類似コマンドとしてwhereisやwhichコマンド

GSIC@t4support:~

File Edit View Help

```
[GSIC@t4support ~]$ find . -name hoge -print
./hoge

[GSIC@t4support ~]$ which python
/usr/bin/python

[GSIC@t4support ~]$ whereis python
python: /usr/bin/python /usr/bin/python3.4 /usr/bin/python3.4m
/usr/bin/python2.7 /usr/bin/python2.7-config /usr/bin/python3.4-config
/usr/bin/python3.4m-config /usr/lib/python3.4 /usr/lib/python2.7
/usr/lib64/python3.4 /usr/lib64/python /usr/lib64/python2.7
/usr/include/python3.4m /usr/include/python2.7 /usr/include/python
/usr/share/man/man1/python.1.gz
```

fileコマンド

- ファイルの型を調べる
- 指定されたファイルの文字列を調べ内容を判断

※判断ミスもあるので、全面的に信用しないこと

GSIC@t4support:~

File Edit View Help

```
[GSIC@t4support ~]$ file hoge
hoge: directory

[GSIC@t4support ~]$cd hoge
[GSIC@t4support hoge]$ file hogedoc
hogedoc: empty

[GSIC@t4support hoge]$ file hogedoc2
hogedoc: ASCII text
```

grepコマンド

- ファイル内の文字列検索
- 標準出力(パイプ)からの文字列検索を行なう

  viエディタやmoreコマンドなどの文字列検索を行なう前に実行すると便利

GSIC@t4support:~

File Edit View Help

```
[GSIC@t4support examples]$ grep mpi *.c
connectivity_c.c:#include <mpi.h>
hello_c.c:#include "mpi.h“

[GSIC@t4support examples]$ grep mpi *.c
connectivity_c.c:#include <mpi.h>
connectivity_c.c:    MPI_Status  status;
(略)
hello_c.c:#include "mpi.h“
hello_c.c:    char version[MPI_MAX_LIBRARY_VERSION_STRING];
(略)
[GSIC@t4support hoge]$ grep -i mpi *.c | more
(略)
```

diffコマンド

2つのファイルの違いを表示する

diff

GSIC@t4support:~

File Edit View Help

```
[GSIC@t4support ~]$ diff hello_c.c hello_cxx.cc
18,20c18,20
<     MPI_Init(&argc, &argv);
<     MPI_Comm_rank(MPI_COMM_WORLD, &rank);
<     MPI_Comm_size(MPI_COMM_WORLD, &size);
---
>     MPI::Init();
>     rank = MPI::COMM_WORLD.Get_rank();
>     size = MPI::COMM_WORLD.Get_size();
```

sdiff

GSIC@t4support:~

File Edit View Help

```
[GSIC@t4support ~]$ sdiff hello_c.c hello_cxx.cc
    MPI_Init(&argc, &argv);                         |     MPI::Init();
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);           |     rank = MPI::COMM_WORLD.Get_rank();
    MPI_Comm_size(MPI_COMM_WORLD, &size);           |     size = MPI::COMM_WORLD.Get_size();
    MPI_Get_library_version(version, &len);             MPI_Get_library_version(version, &len);
```

オンラインマニュアル

- man コマンド名/ファイル名
- man -k キーワード

GSIC@t4support:~

File Edit View Help

```
[GSIC@t4support examples]$ man ls
Man: find all matching manual pages (set MAN_POSIXLY_CORRECT to avoid this)
 * ls (1)
   ls (1p)
Man: What manual page do you want?
Man:
NAME
       ls - list directory contents
SYNOPSIS
       ls [OPTION]... [FILE]...
DESCRIPTION
       List information about the FILEs (the current directory by default).  Sort
entries alphabetically if none of -cftuvSUX nor --sort is specified.
       Mandatory arguments to long options are mandatory for short options too.
```

POSIX
【Portable Operating System Interface for UNIX】
IEEEによって定められた、UNIXベースのOSが備えるべき最低限の仕様のセット。
(e-wordより抜粋)

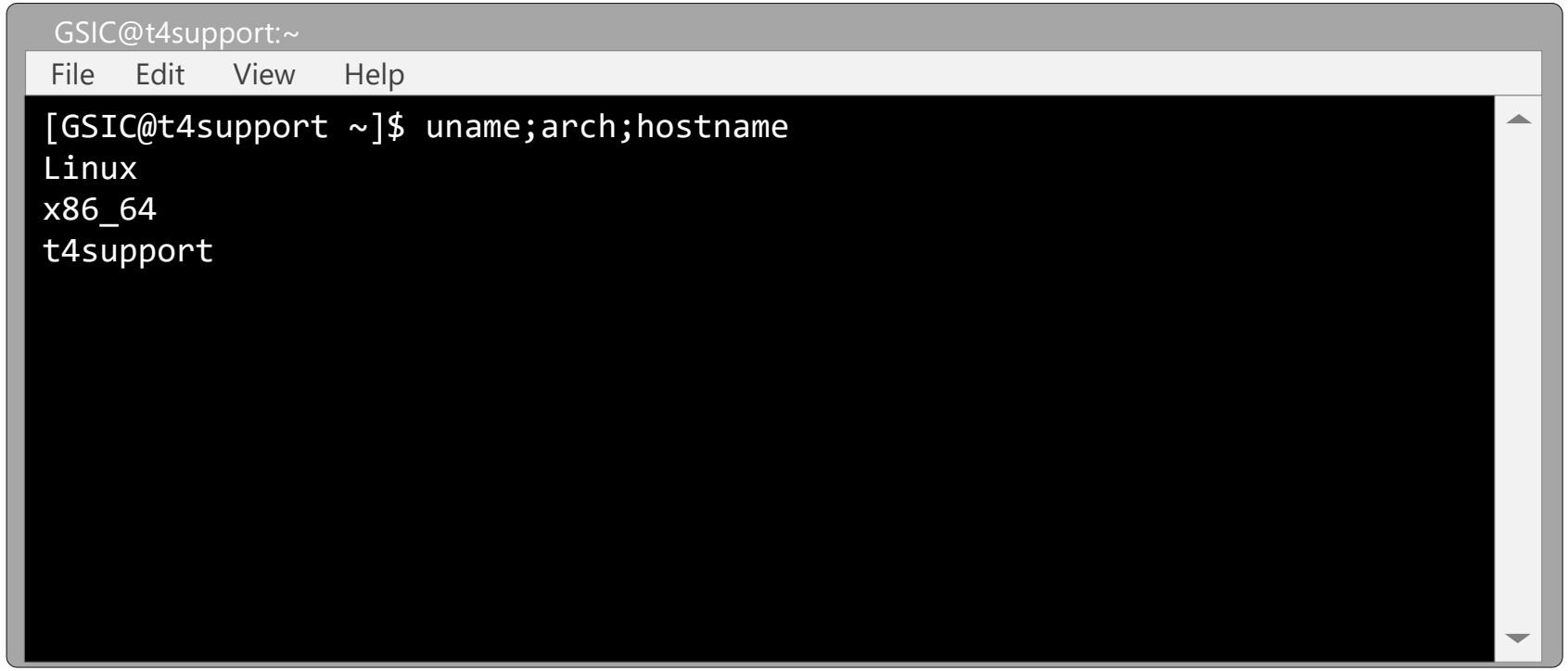
コマンドの結合

コマンドを結合させて、プログラムのように一括処理ができる

- コマンドをパイプ(|)でつなぐ
- コマンドの実行結果をファイルに書き出す
- ファイルをコマンドに入力する

セミコロン

- 複数のコマンドをセミコロンでつなぐ(グルーピング)
- コマンド1の実行後に、コマンド2,コマンド3が順次実行される



パイプ

- 例)catの標準出力をgrepに渡す

GSIC@t4support:~

File Edit View Help

```
[GSIC@t4support ~]$ cat /usr/mpi/gcc/openmpi-1.10.4/include/mpi.h | grep INDEX
#define MPI_T_ERR_INVALID_INDEX       57
  MPI_COMBINER_INDEXED,
  MPI_COMBINER_HINDEXED_INTEGER,
  MPI_COMBINER_HINDEXED,
  MPI_COMBINER_INDEXED_BLOCK,
  MPI_COMBINER_HINDEXED_BLOCK
```

リダイレクション

- > の例　　lsの結果を`ls.txt`として作成
- >>の例　　lsの結果を`ls.txt`に追記

GSIC@t4support:~

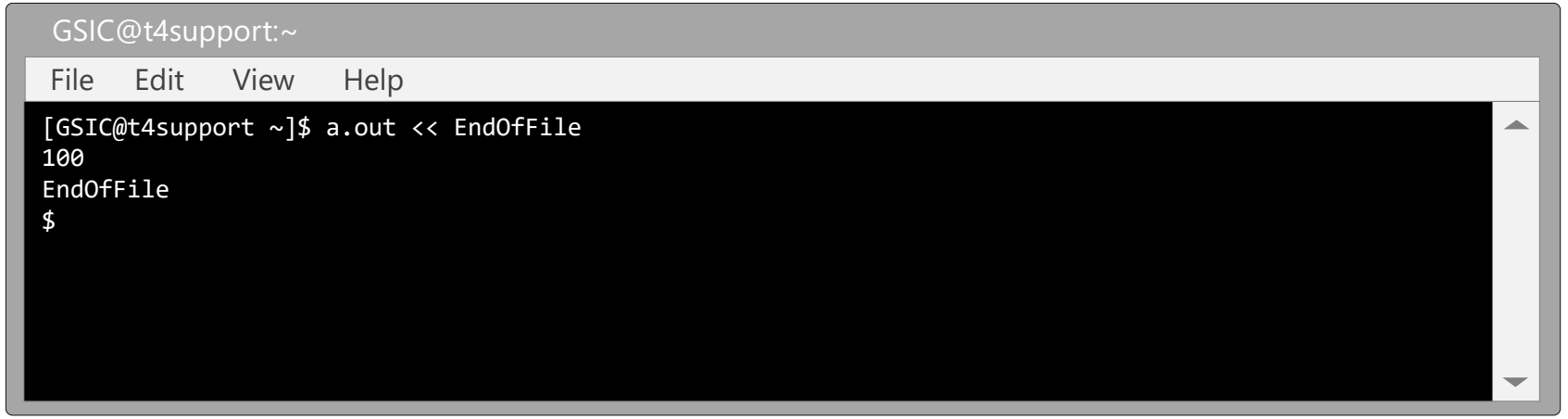
File Edit View Help

```
[GSIC@t4support ~]$ ls
Desktop hoge
[GSIC@t4support ~]$ ls > ls.txt
[GSIC@t4support ~]$ cat ls.txt
Desktop
hoge
[GSIC@t4support ~]$ ls >> ls.txt
[GSIC@t4support ~]$ cat ls.txt
Desktop
hoge
Desktop
hoge
ls.txt
```

ヒアドキュメント

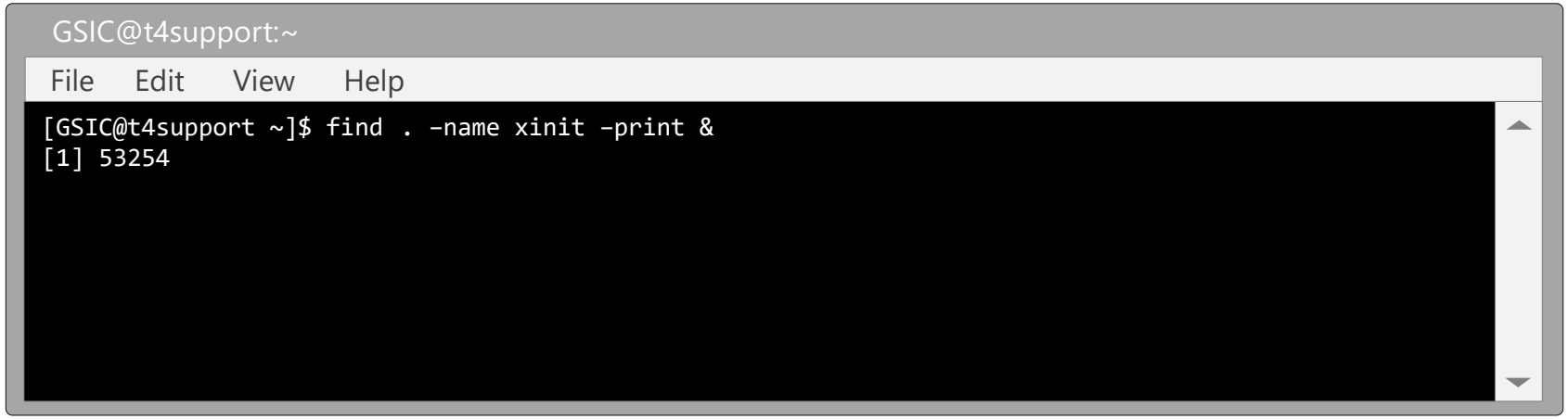
- 標準入力からデータをコマンド(プログラム)へ渡す
  - ファイルからの入力

  - 標準入力からの入力



バックグラウンド実行

- 実行時間の大きなコマンド(プログラム)
- コマンド実行中に他の事をしたい
  →バックグラウンドジョブとして実行
  ※フォアグラウンドでコマンドの実行は問題ない
- コマンドの後ろに&をつけて実行



プロセス

- プロセスはOS上の処理の実行単位のこと
  - 下の例では3プロセス

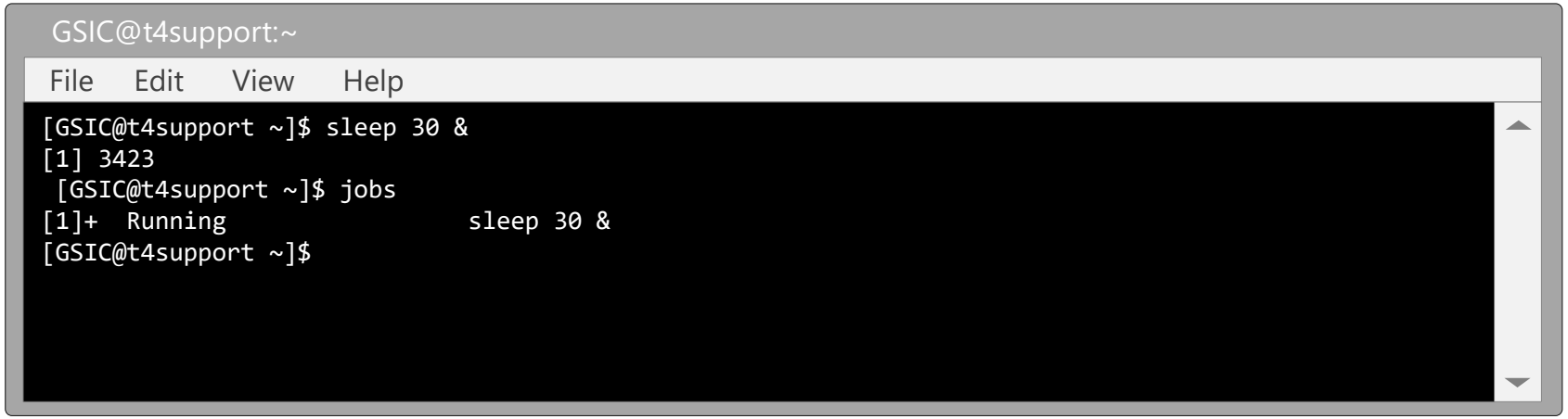
GSIC@t4support:~

File Edit View Help

```
ux00000@r6n2:~> top
  PID USER      PR  NI    VIRT    RES    SHR S   %CPU  %MEM     TIME+ COMMAND
354797 hpe_use+  20   0 49.897g 456836 408116 R 14.286 0.173   0:02.09 pmemd.cuda.MPI
354798 hpe_use+  20   0 49.897g 453680 407172 R 14.286 0.172   0:02.08 pmemd.cuda.MPI
  3207 hpe_use+  20   0  425428  60592   1356 S  2.640 0.092 30:26.49 a.out
```

ジョブ

- ジョブはコマンド/プログラムをまとめたシェルの実行単位のこと
  - コマンドをパイプなどでつなげた場合もジョブ
  - jobsコマンドで実行ジョブを確認できる



練習

- ここまでに出てきたコマンドを練習として使ってみましょう

シナリオ：研究室から以下の指示をされました。
グループディスクにおいているバッチスクリプトを
ディレクトリごとホームディレクトリにコピーして修正する。

```
cd
mkdir lesson
cd lesson
cp –r /gs/bs/soudan/UNIX/* .
cp sample.sh sample.txt
file sample.txt
vi sample.txt
```

- iもしくはaキーを押し入力モードにし、どこかを適当に編集し,ESCキーを押し(待機状態)、:wqとタイプ(保存終了)してください。

```
diff sample.sh sample.txt
sdiff sample.sh sample.txt
```

解説

```
cd                                  ＃ホームに移動
mkdir lesson                        ＃lessonディレクトリを作成
cd lesson                           ＃lessonディレクトリに移動
cp /gs/bs/soudan/UNIX/* .       ＃UNIX以下のファイルをコピー
cp sample.sh sample.txt             ＃sample.shをsample.txtとしてcp
file sample.txt                     #sample.txtの型を確認
vi sample.txt                       ＃sample.txtの修正
diff sample.sh sample.txt           ＃比較
sdiff sample.sh sample.txt
```

プログラムの実行

- moduleコマンドについて
- バッチジョブ

moduleコマンドについて

- 利用するソフトウェアに関係する環境設定を、前もってmoduleコマンドで行う
  - 例： $ module load intel　　Intelコンパイラ
    - module load intel/2024.0.2のようにバージョン指定も可能
  - 例： $ module load gnuplot

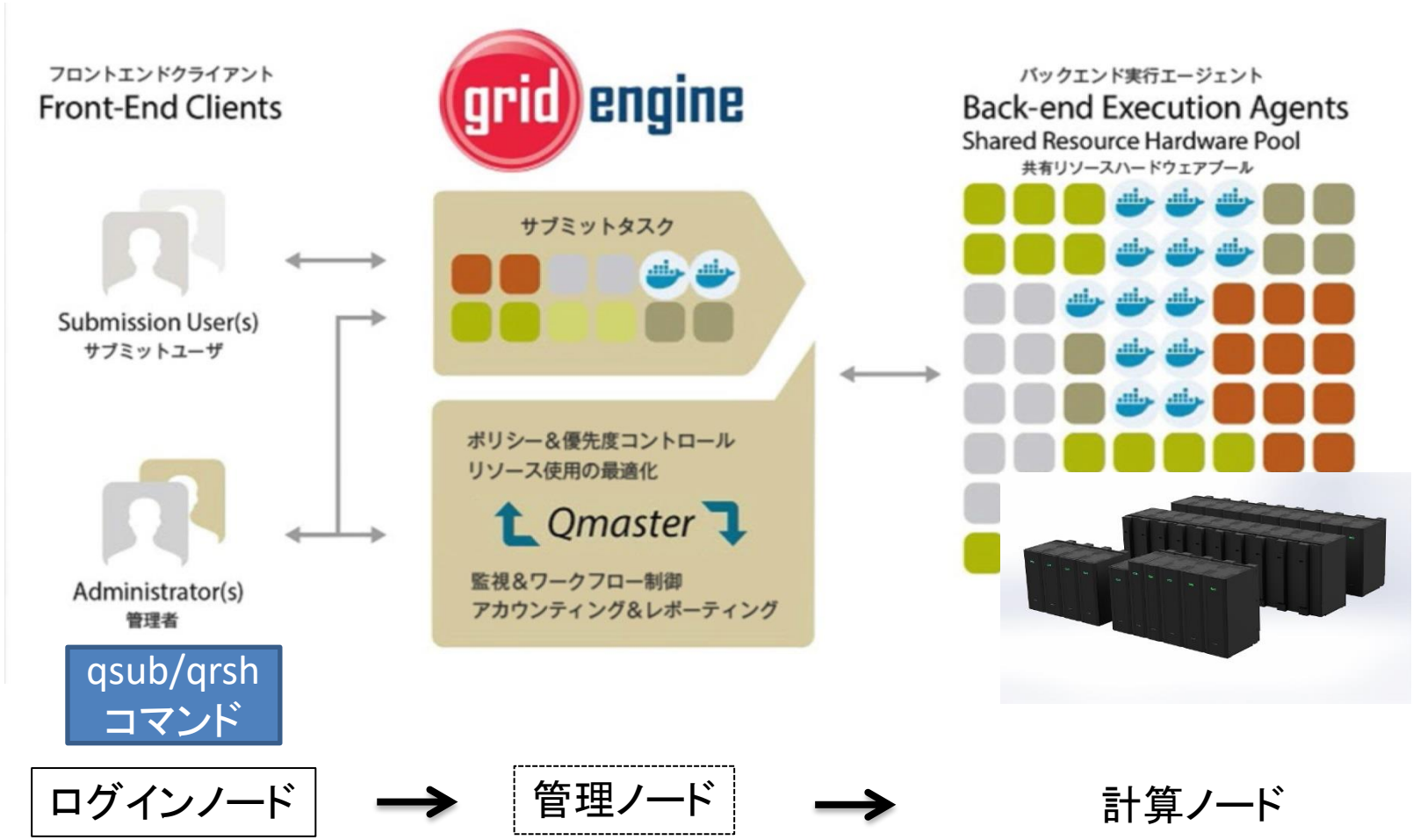
- 用意されているモジュール一覧： $ module avail
- モジュールによって、依存モジュールが自動読込される
  - たとえば、openmpi/5.0.2-gccモジュールはintel-mpiモジュールに依存

- よく使うmodule option

コマンドオプション	効果
module avail	準備されている環境の確認
module load	環境の呼び出し
module list	呼び出した環境の確認
module purge	呼び出した環境の破棄

バッチジョブ

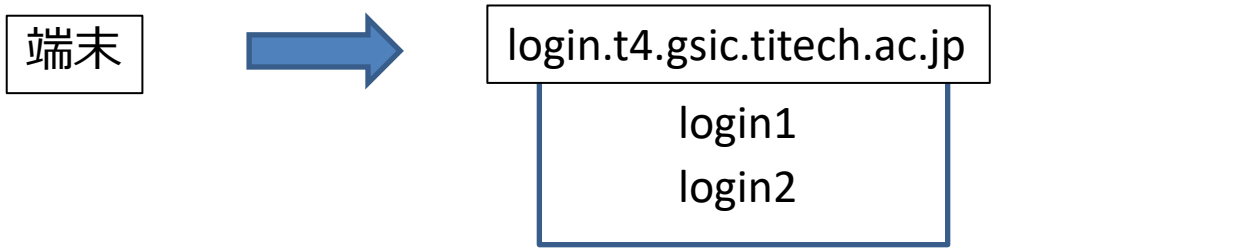
- ジョブスケジューラにAGEが導入されており、計算資源を管理



計算ノード利用イメージ (コマンド)

- ログイン(SSH鍵認証のみ)
  - 研究室内PCなどより，sshコマンドを実行．負荷分散を考慮し，自動的にログインノードのいずれかにログインされる．

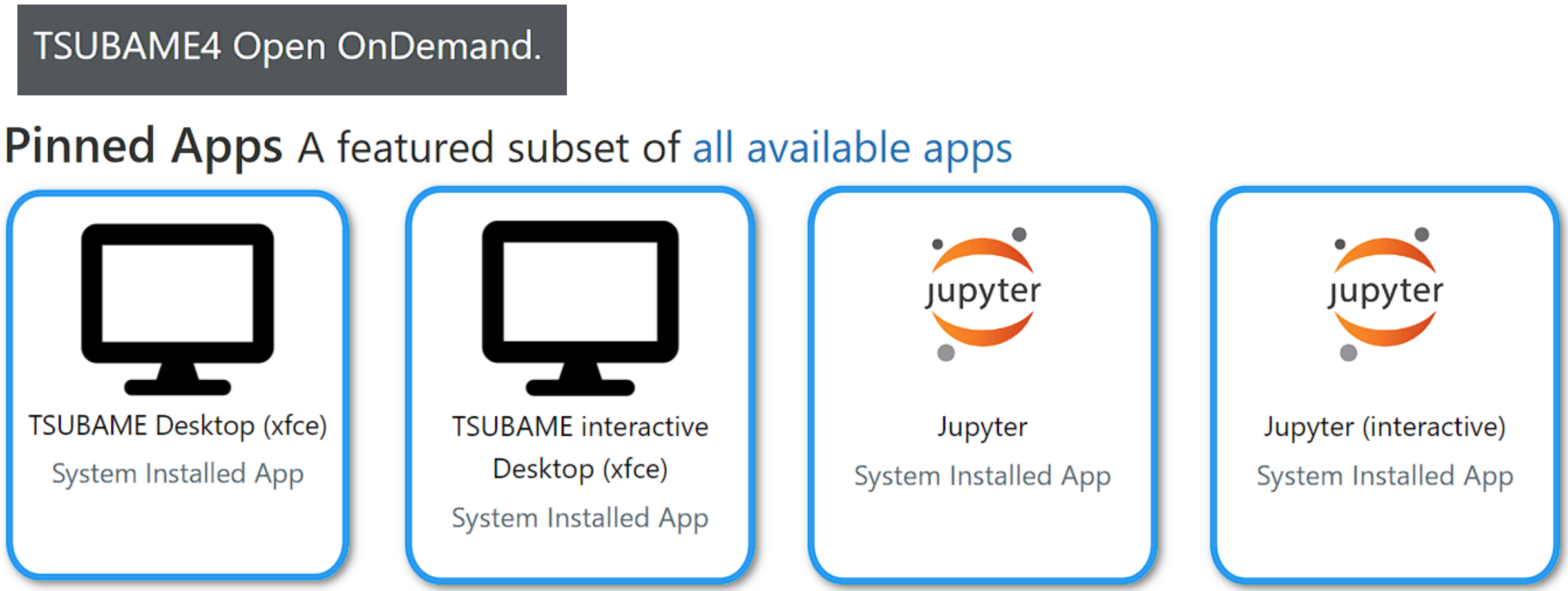
「TSUBAME4.0ポータル利用説明書」参照



負荷分散によりloginノードのどちらかにログイン

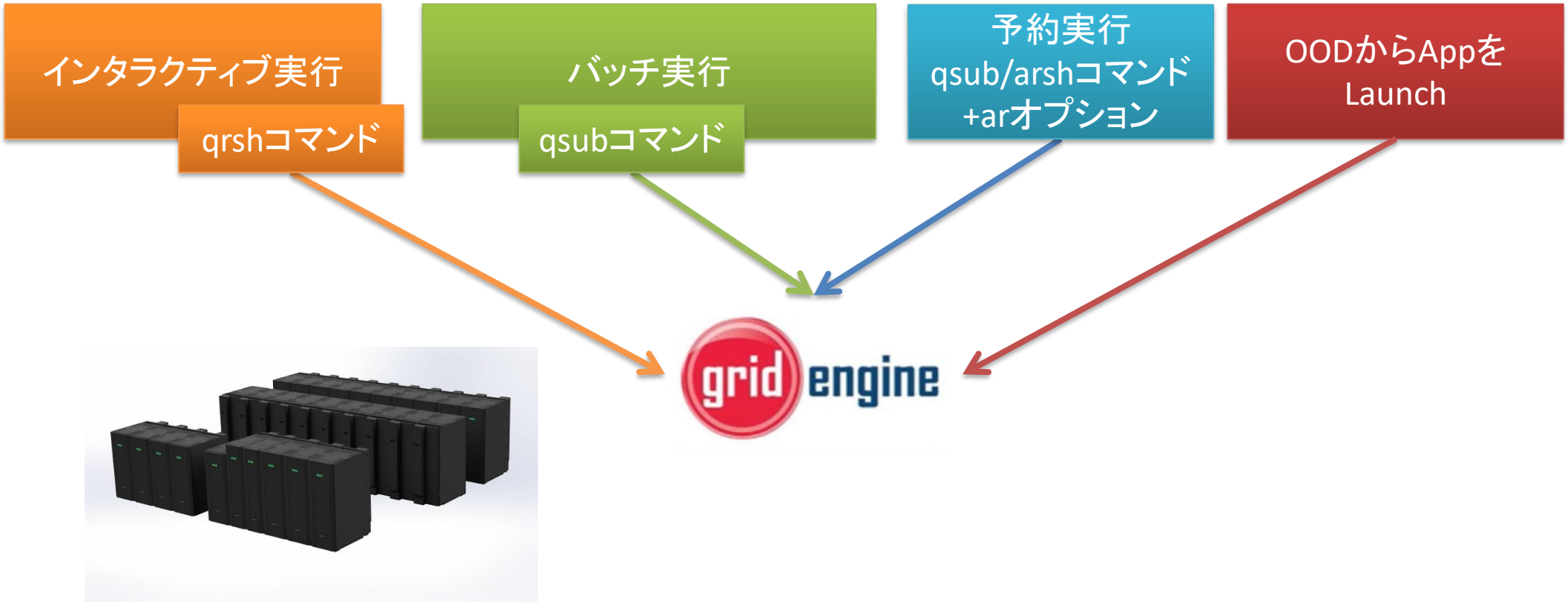
TSUBAME4 Open OnDemand

- OODへログインしAppをLaunchすることでも計算ノードを利用できる。



計算ノード利用イメージ (2)

ジョブはqrsh/qsubコマンドで240ノードある計算ノードのいずれかで実行される
ユーザによるノード指定は基本的に出来ない
インタラクティブ実行かバッチ実行(通常/予約)、T4ではOOD



利用に際して

- ユーザはプログラムを用意する
- その実行に必要な計算機の規模・台数を考える
- 「資源タイプ」の選択を行う。
  - node_f (ノード一台まるごと)、node_q (1/4)、cpu_4 (4CPUコアだけ)…
- 資源タイプ・動かしたいプログラムの情報をもとに「ジョブスクリプト」ファイルを作る
- qsubコマンドにより、ジョブの投入を行う
- qsubなどのコマンドは、ログインノードやインタラクティブノードで実行可能
- 結果が出るまで待つ

資源タイプの構成

- 下記の資源タイプが存在し、SSH接続が可能なのはnode_fのみ
- コマンドオプションとしては-lで資源タイプ名を指定する

資源タイプ	使用物理CPUコア数	メモリ (GB)	GPU 数	ローカルスクラッチ領域 (GB)
node_f	192	768	4	1920
node_h	96	384	2	960
node_q	48	192	1	480
node_o	24	96	1/2	240
gpu_1	8	96	1	240
gpu_h	4	48	1/2	120
cpu_160	160	368	0	96
cpu_80	80	184	0	48
cpu_40	40	92	0	24
cpu_16	16	36.8	0	9.6
cpu_8	8	18.4	0	4.8
cpu_4	4	9.2	0	2.4

利用料金は「大きい」資源ほど高いので、必要十分な資源を選ぶのがおすすめ

- MPIジョブ等では、node_f×4、cpu_16×10のように複数個の資源を利用可能
  – 異種混在は不可
- なお「インタラクティブノード」はnode_o相当を、複数ユーザで共有

計算ノード利用の制限事項

- 制限事項の詳細は各種制限値一覧をご確認ください

  - 1ジョブの利用可能時間
    - 最大24時間(予約は条件により1週間程度利用可能)

  - 1ユーザあたり同時実行可能なスロット数(CPUコア数)
    - 一人あたり6144(週末12288)スロット

  - 1ジョブあたりの最大並列度
    - 64並列

    実装上の理由により64並列としているが、6144スロットの制限があるためnode_fでは32並列が最大となることに注意

  - 1ユーザあたり同時実行可能なジョブ数
    - 一人あたり最大30(週末100)ジョブ

プログラムの実行（シェルの作成）

- バッチで実行するためのジョブスクリプトの作成

詳細はTSUBAME4.0利用の手引きを参照

```
#!/bin/bash                 <-シェバン、スクリプトのインタプリタ指定
#$ -cwd                     <-カレントドライブで実行するためのバッチジョブスケジューラー用オプション
#$ -N test_job        <-ジョブ名を指定
#$ -l cpu_4=1             <-資源タイプを指定、s_core(1cpu)を1つ
#$ -l h_rt=0:10:0          <-計算ノードを確保する時間を指定、10分

echo "this host is" `hostname` "."
```

プログラムの実行（バッチ実行）

- qsub -l 資源名 -l 確保時間 -g グループ ジョブスクリプト

詳細はTSUBAME4.0利用の手引きを参照

GSIC@t4support:~

File Edit View Help

```
[ux00000@login1 ~]$ qsub -g グループ sample.sh

#お試し実行の場合は10分、2ノード以内であることを確認し、-gオプション無しで実行する
[ux00000@login1 ~]$ qsub sample.sh

#資源タイプnode_fでインタラクティブ実行した場合
[ux00000@login1 ~]$ qrsh -g GSIC -l node_f=1 -l h_rt=8:0:0
```

バッチキューの確認

- ジョブが投入されていることを確認

GSIC@t4support:~

File Edit View Help

```
ux00000@login1:~> qstat
job-ID     prior   name       user         state submit/start at     queue    jclass slots ja-task-ID
-----------------------------------------------------------------------------------------------------
93501      0.55500 PDF        ux00000   r    10/04/2017 07:39:58 all.q@r2i4n6    56
93578      0.55500 QRLOGIN    ux00000   r    10/04/2017 11:39:58 all.q@r6n2    28
```

- ジョブを消去

GSIC@t4support:~

File Edit View Help

```
ux00000@login1:~> qdel 93578
ux00000@login1:~> qstat

ux00000@login1:~>
```

プログラムの実行サイズ

- 実行時のメモリサイズの調べ方
- ※topの場合は確認次第qキーを押してtopを停止する。

GSIC@t4support:~

File Edit View Help

```
ux00000@r6n2:~> ps aux
USER              PID %CPU %MEM    VSZ   RSS    TTY   STAT START   TIME COMMAND
ux00000     354797 13.2  0.1 52321684 490304 pts/1 Rl   13:19   0:02 pmemd.cuda.MPI
ux00000     354798 13.2  0.1 52321348 487148 pts/1 Rl   13:19   0:02 pmemd.cuda.MPI

ux00000@r6n2:~> top
  PID USER      PR  NI    VIRT    RES    SHR S   %CPU  %MEM     TIME+ COMMAND
354797 hpe_use+  20   0 49.897g 456836 408116 R 14.286 0.173   0:02.09 pmemd.cuda.MPI
354798 hpe_use+  20   0 49.897g 453680 407172 R 14.286 0.172   0:02.08 pmemd.cuda.MPI
```

プログラムの強制終了

- 実行中のジョブのプロセス番号の確認

  例. psコマンドによる確認/ Topコマンドによる確認

GSIC@t4support:~

File Edit View Help

```
ux00000@r6n2:~> ps aux | grep ux00000
354797 13.2  0.1 52321684 490304 pts/1 Rl   13:19   0:02 pmemd.cuda.MPI -O -i input -p top -o test
354798 13.2  0.1 52321348 487148 pts/1 Rl   13:19   0:02 pmemd.cuda.MPI -O -i input -p top -o test

ux00000@r6n2:~> top
354797 hpe_use+  20   0 49.897g 456836 408116 R 14.286 0.173   0:02.09 pmemd.cuda.MPI
354798 hpe_use+  20   0 49.897g 453680 407172 R 14.286 0.172   0:02.08 pmemd.cuda.MPI
```

- Kill コマンドの実行

GSIC@t4support:~

File Edit View Help

```
ux00000@r6n2:~> kill 354797 354798

# 上記コマンドで消えない場合
ux00000@r6n2:~> kill -9 354797 354798
```

処理情報収集コマンド(1)

- ログインしているユーザーを表示(whoコマンド)
- ログインユーザーの状態を表示(wコマンド)

GSIC@t4support:~

File Edit View Help

```
[ux00000@t4support ~]$ who
GSIC     pts/5        2017-09-06 10:52 (:1)
GSIC     pts/6        2017-09-06 11:22 (:1)
TEST     pts/2        2017-09-07 10:05 (192.168.111.63)
GSICUSE pts/7        2017-10-03 12:30 (:2)
GSICUSE pts/3        2017-10-04 12:37 (192.168.111.63)


[ux00000@t4support ~]$ w
 12:39:20 up 51 days,  3:29, 12 users,  load average: 1.00, 1.01, 1.00
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
GSIC     pts/5    :1               0611月10 ?xdm?  22days 22.80s gdm-session-worker
GSIC     pts/6    :1               0611月10 41days  0.25s 10:00  /usr/libexec/
TEST     pts/2    192.168.111.63   06 9月17 4days  1.26s  1.26s -bash
GSICUSE pts/7    :2               火12 24:09m  0.05s  0.05s bash
GSICUSE pts/3    192.168.111.62   12:37 0.00s  0.04s  0.00s w
```

処理情報収集コマンド(2)

- 実行jobを表示(topコマンド)

GSIC@t4support:~

File Edit View Help

```
[GSIC@t4support ~]$ top
Tasks: 784 total,   1 running, 781 sleeping,   2 stopped,   0 zombie
%Cpu(s):  0.1 us,  0.1 sy,  0.0 ni, 99.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:  26377451+total, 14203016 used, 24957150+free,     3352 buffers
KiB Swap:        0 total,        0 used,        0 free. 10473748 cached Mem
   PID USER      PR  NI    VIRT    RES    SHR S   %CPU  %MEM     TIME+ COMMAND
225488 root      20   0  371260  20796   4816 S  2.990 0.008 118:58.30 nv-hostengine
  1111 root      20   0  110076  66204  61604 S  0.332 0.025   0:19.26 systemd-journal
```

- 実行jobを表示(psコマンド)

GSIC@t4support:~

File Edit View Help

```
[GSIC@t4support ~]$ ps aux

[GSIC@t4support ~]$ ps aux | grep
ps aux | grep 1111
root        1111  0.0  0.0 110076 66224 ?        Ss   Sep28   0:19 /usr/lib/systemd/systemd-journald
GSIC      352412  0.0  0.0  10240  1640 pts/0    S+   12:29   0:00 grep --color=auto 1111
```

利用環境

- シェル

シェル

- コマンドを実行する事が出来る環境をシェルと呼ぶ
- 環境は、ユーザの好みに合わせて変更できる
- TSUBAME4でサポートしている(chsh可能な)シェルは以下のとおり
  - /bin/bash
  - /bin/tcsh
  - /bin/zsh
- 下記のようにフルパスで変更したいシェルを指定してください
  - $ chsh /bin/tcsh

  変更の反映は5分程度かかります

環境

- デフォルトのログインシェルはbash(chshで変更可能)
- 起動時に確認する設定ファイルの順番

  /etc/profileおよび/etc/bashrc

  ~/.bash_profile

  ~/.bash_login(~/.bash_profileがなければ)
- 対話ログインは、~/.bashrcが読み込まれる
- 個人設定ファイル~/.bashrcの編集
  - PATH="/usr/games:$PATH“
  - export PATH
- 編集後に有効にするには再ログインか source .bashrcとする
- aliasもここに書いておくとよい。
- 変更する場合は、良く確認してから更新してください
- また、新しいターミナルで必ず確認してください

変数

- シェル変数
  - プロセス内でのみ有効な変数
- 環境変数
  - プロセス内に限らず子プロセスにも渡る変数

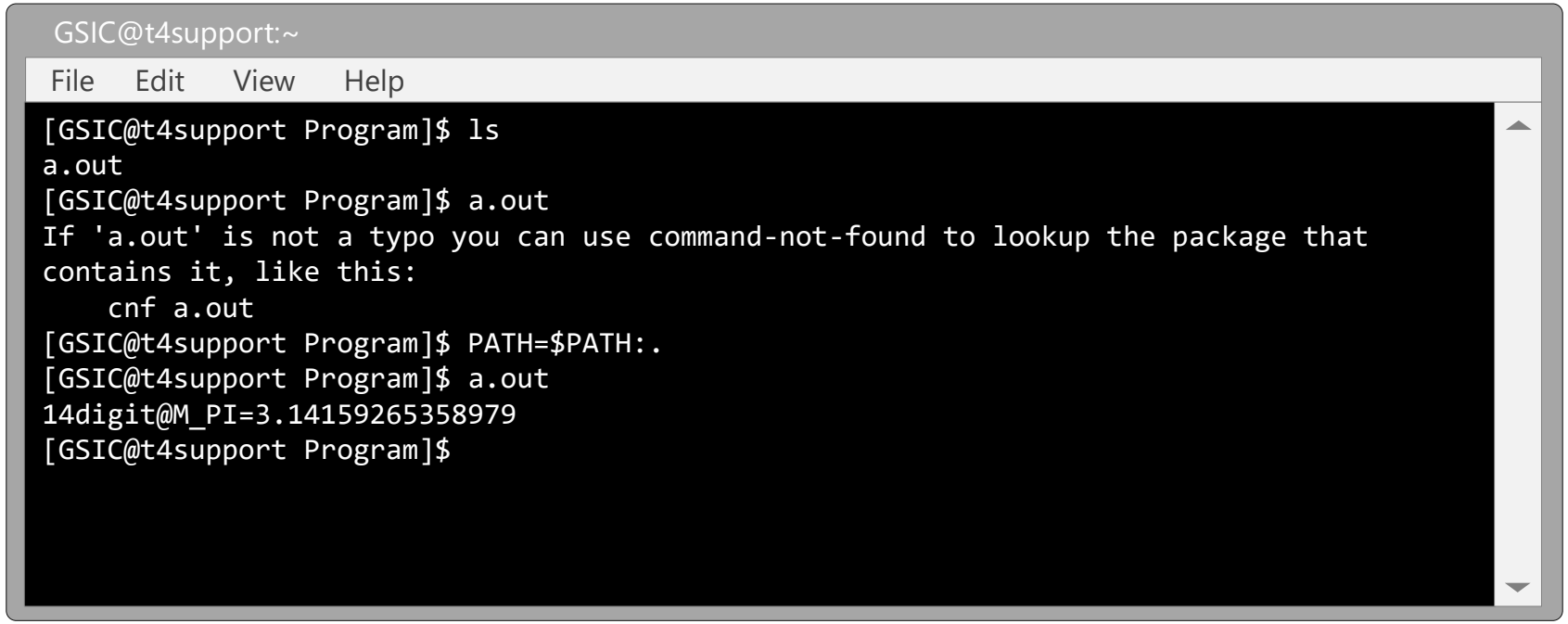
HISTSIZEシェル変数

- History機能
  - 今までに使用していたコマンド履歴を決められた数だけ保存。
  - コマンド実行時に有効に再利用できるようにする。
  - 保存したいコマンド履歴の数は、任意の数を選べる。
  - コマンド履歴は、この機能を有効にしてから保持。
- 設定例
  - $ export HISTSIZE=600 ⇒ コマンド履歴サイズを600にする
  - $ history 3 ⇒ ヒストリリストを3件表示する

```
534 cd
535 ls
536 history 3
```

PATH変数

- コマンド等の検索場所を設定
- 独自に作成したコマンド、プログラム等の保存場所を指定



※ .bashrc 等に記述しておけば毎回有効になる

ファイル転送

- ネットワークで接続されたシステムをホストと呼ぶ。
- このホスト間では、簡単に他のホストのファイルをコピーできる。
- この機能を使用するためのコマンド
  - ftp,rcp,rsync,sftp,scp
- 本学ではrsync,sftp,scpを導入

rsync/sftp/scpコマンドの例

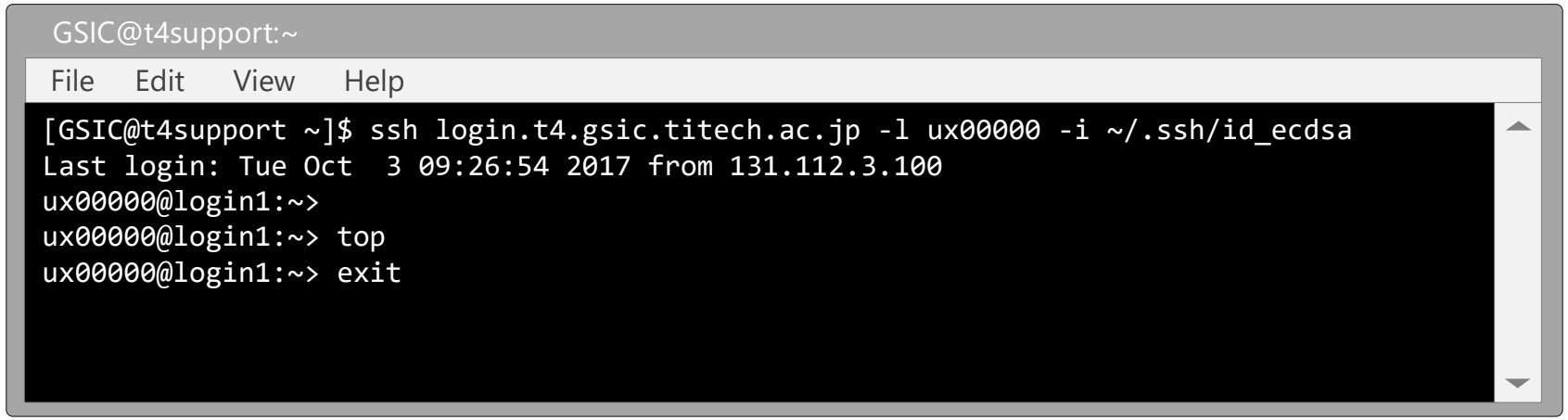
GSIC@t4support:~

File Edit View Help

```
#rsync
[GSIC@t4support ~]$ rsync -av --progress -e "ssh -i .ssh/id_ecdsa -l ux00000"
login.t4.gsic.titech.ac.jp:/gs/bs/soudan/UNIX/testfile ./
receiving incremental file list
testfile
         990 100%  966.80kB/s    0:00:00 (xfer#1, to-check=0/1)

sent 42 bytes  received 1078 bytes  2240.00 bytes/sec
total size is 990  speedup is 0.88
[GSIC@t4support ~]$

#sftp
[GSIC@t4support ~]$ sftp -i ~/.ssh/id_ecdsa ux00000@login.t4.gsic.titech.ac.jp
Connected to login.t4.gsic.titech.ac.jp.
sftp> get /gs/bs/soudan/UNIX/testfile
Fetching /gs/bs/soudan/UNIX/testfile to testfile
/gs/hbs/soudan/UNIX/testfile
sftp> exit

#scp
[GSIC@t4support ~]$ scp -i ~/.ssh/id_ecdsa
ux00000@login.t4.gsic.titech.ac.jp:/gs/bs/soudan/UNIX/testfile .
testfile
```

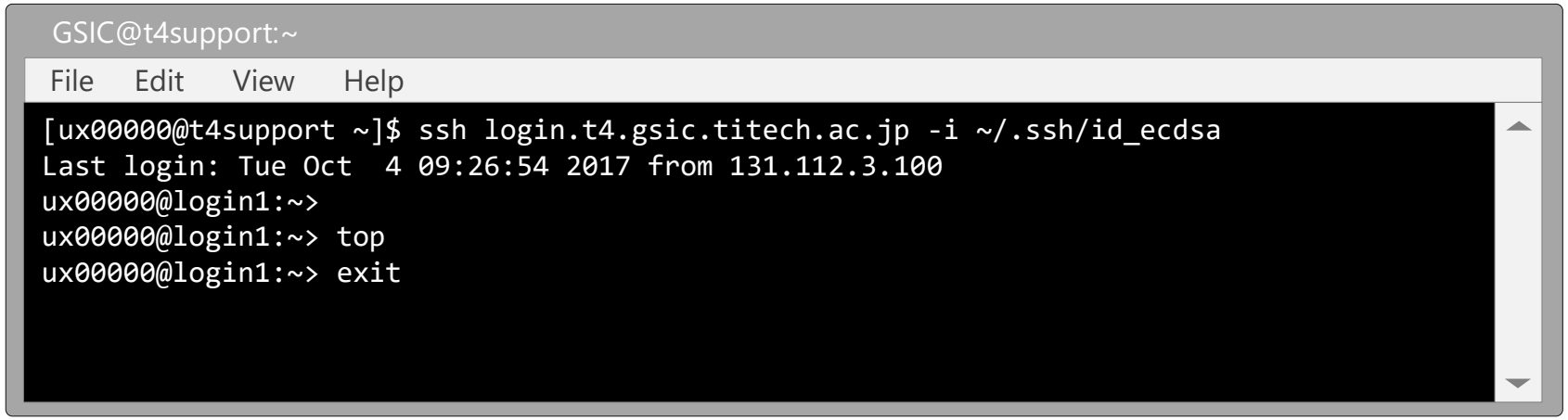
sshコマンドの例

- ログイン名を指定してのログインGSIC→GSCIUSER00



- 使用マシンのアカウントのログイン名がTSUBAMEと同じ場合のログインではユーザ名を省略可

  ux00000→ ux00000



X-Window (X)システムの特徴

- Linuxをはじめ多くのOSで使用
- GUIアプリケーションを構築
- X、X11などの別名あり
- ハードウェアに依存しない
  - Linux標準装備
  - Windows用のソフトも多数
    - MobaXterm
    - PuTTy・Teraterm・RLogin + VcXsrv/X-ming
    - Cygwin
  - MacintoshはXQuartzが別途必要

クライアント/サーバ

- Xアプリケーションの動作
  → クライアントプログラム、サーバプログラム

- Xプロトコルという独自の通信手順
  → クライアント/サーバ間のデータのやり取り
    - サーバ側（表示される側）
      どのシステムからの表示要求を許すかを定義
      % xhost +クライアントホスト名

    - クライアント側（表示させる側）
      DISPLAY環境変数を指定
      % export DISPLAY=サーバホスト名

TSUBAME上でXを利用する

- 以下の手順にてXが利用可能です。
  ユーザ環境によってはXがうまく動かない場合がありますのでその場合はOODの利用を検討することも可能です。

  – qrshオプションで実行する方法
    $ qrsh –g TSUBAME4グループ -l cpu_4=1,h_rt=確保時間

  – 別ターミナルを利用する方法(node_fのみ)
    $ qrsh -l node_f=1 -l 確保時間 -g TSUBAME4グループ
    確保時間はhh:mm:ss表記でも秒表記でもOK(24:00:00 や 600)
    例:qrsh -l node_f=1,h_rt=600 -g tga-hogehoge
    ユーザ名@**割り当てられた計算ノード**
    *別ターミナルを立ち上げてもOK
    ssh login.t4.gsic.titech.ac.jp -YC
    ssh **割り当てられた計算ノード** -YC