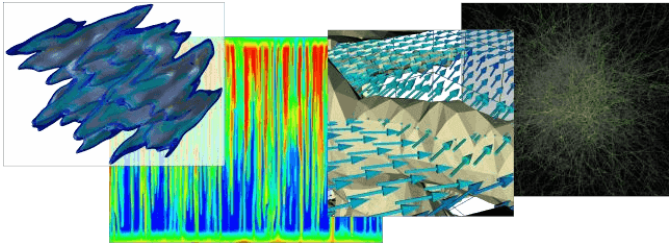
Introducción al emulador de terminal

Parte I: `bash`, `g++`, `cmake`, `gnuplot`, `gmsh`, `paraview`



Practique los ejemplos en Gitpod

Open in Gitpod

Disponible en

¡Únete al grupo en Telegram!

GNU/Linux es un sistema operativo de código abierto. Este fue diseñado para ser similar a UNIX. UNIX Linux se refiere al kernel. Linux realmente no es un UNIX.

Historia del desarrollo de los sistemas Unix

- La primera versión de Unix fue escrito por Dennis Ritchie y Ken Thompson en 1969.
- Durante los 1970s, los laboratorios Bell no permitían vender Unix comercialmente, sin embargo, lo distribuyeron a universidades a bajo costo.
- La Universidad de California en Berkeley añadió sus propias implementaciones, el resultado es conocido como la versión BSD de Unix.
- Tempranamente en 1980s, AT&T firmó un acuerdo en el que permitía vender System III, la primera versión Unix de AT&T.
- Su popularidad se incrementó en esta década, varias compañías modificaron el código de Unix para crear sus propias variantes. Ejemplos son SunOS, Ultrix, HP-UX.
- Estas variantes fueron similares, pero a su vez existían ligeras diferencias en el conjunto de cabeceras y listas diferentes de funciones en las bibliotecas del sistema.
- El estándar POSIX ayudó a eliminar estas diferencias, sin embargo, POSIX trajo nuevas características en diferentes tiempos, conllevando a nuevas variantes.
- Los autores querían que sus programas corran en una amplia variedad de Unix, la idea general fue usar `#ifdef`, sin embargo, incrementaba la dificultad de conocer qué versiones tenían ciertas características.

Empezando con GNU/Linux

Nos otorga un entorno de programación con un programa base llamado shell que nos permite comunicarnos entre el kernel y las aplicaciones de usuario, por lo que controlamos el sistema operativo. Tiene un lenguaje de scripting que nos permite automatizar tareas que sigue el estándar POSIX.

- Abrir terminal Ctrl + Alt + t.
- Obtener los comandos digitados recientemente ↑ / ↓.
- Limpiar la consola Ctrl + l.
- Moverse entre palabras Alt + f / Alt + b.
- Moverse al inicio/final del comando Home / End.
- Buscar comandos utilizados anteriormente Ctrl + r / Ctrl + s.
- Interrumpir la ejecución de un programa Ctrl + c.
- Enviar a segundo plano un proceso Ctrl + z.
- Autocompletar el nombre de un archivo ⇤⇥.
- Despliega el número de posibilidades para el comando ⇤⇥ + ⇤⇥.

- Linux comenzó como un proyecto de pasatiempo por Linus Torvalds en 1991.
- La fuente se hizo disponible libremente y otros se unieron para formar este sistema operativo de vanguardia.
- Como fue creado desde cero, los primeros usuarios pudieron influir en el rumbo del proyecto y asegurarse de no cometer los mismos errores de otros UNIXes.
- Los proyectos de software, como Dune Project, toman la forma de código fuente.
- Es un conjunto de instrucciones de cómputo legibles para el humano.
- Linux está siendo escrito principalmente en C.

Comandos básicos

Existen algunas convenciones acerca del directorio de trabajo, tiene la forma `/foo/bar`.

```
[user@host somedir]$ pwd
/home/user/somedir
```

Cada directorio tiene por lo menos dos directorio, el directorio padre `..` y el directorio actual `.` como se ve a continuación.

```
[user@host somedir]$ ls -a
. ..
```

Cuadro: Comandos básicos

Comando	Utilidad
`pwd`	Muestra la ruta absoluta del directorio actual.
`cd`	Navega al home del usuario.
`ls`	Lista los archivos del directorio en el que se encuentra.
`cp origen destino`	Copia un archivo origen al destino.
`mkdir nombre`	Crea una carpeta vacía llamada nombre.
`find /foo/bar -name '*.cc'`	Busca archivos con extensión `.cc` en la ruta `/foo/bar`.
`chmod +x ejemplo`	Dar permisos de ejecución al binario ejemplo.
`rm archivo`	Elimina el inodo del archivo.
`cat archivo`	Imprime el contenido de un archivo a la salida estándar.

```
gitpod ~/dune-basics $ cd
```

Classes

Listado 1: Programa `hello-linux.cc`.

```
// Tomado de https://stackoverflow.com/a/66161001

#include <sys/utsname.h>
#include <iostream>

// un pequeño ayudante para mostrar el contenido de una estructura utsname:
std::ostream &operator<<(std::ostream &os, const utsname &u)
{
  return os << "sysname : " << u.sysname << '\n'
            << "nodename: " << u.nodename << '\n'
            << "release : " << u.release << '\n'
            << "version : " << u.version << '\n'
            << "machine : " << u.machine << '\n';
}

int main()
{
  utsname result; // declarar la variable para contener el resultado

  uname(&result); // llamar a la función uname() para completar la estructura

  std::cout << result; // mostrar el resultado usando la función ayudante
}
```

Classes

Listado 2: Programa `dune-basics.cc`.

```
#ifdef HAVE_CONFIG_H
#include "config.h"
#endif
#include <iostream>
#include <dune/common/parallel/mpihelper.hh> // An initializer of MPI
#include <dune/common/exceptions.hh>         // We use exceptions

int main(int argc, char **argv)
{
  try
  {
    // Maybe initialize MPI
    Dune::MPIHelper &helper = Dune::MPIHelper::instance(argc, argv);
    std::cout << "Hello World! This is dune-basics." << std::endl;
    if (Dune::MPIHelper::isFake)
      std::cout << "This is a sequential program." << std::endl;
    else
      std::cout << "I am rank " << helper.rank() << " of " << helper.size()
                << " processes!" << std::endl;
    return 0;
  }
  catch (Dune::Exception &e)
  {
    std::cerr << "Dune reported error: " << e << std::endl;
  }
  catch (...)
  {
    std::cerr << "Unknown exception thrown!" << std::endl;
  }
}
```

Classes

Listado 3: Programa `dune-basics.cc`.

```
#include <iostream>
#include <dune/common/math.hh>

int main(int argc, char **argv)
{
  std::cout << Dune::StandardMathematicalConstants<double>::pi()
            << "\n"
            << Dune::StandardMathematicalConstants<double>::e()
            << "\n"
            << Dune::factorial<int>(3)
            << "\n"
            << Dune::factorial(5)
            << "\n"
            << Dune::power<int, int>(2, 3)
            << "\n"
            << Dune::binomial(100, 1)
            << "\n"
            << Dune::conjugateComplex(std::complex<double>(0, 1))
            << "\n"
            << Dune::sign<double>(24)
            << "\n"
            << Dune::isFinite(std::complex<double>(0, 1))
            << "\n"
            << Dune::isInf(std::complex<double>(0, 1))
            << "\n"
            << Dune::isNaN(std::complex<double>(0, 1))
            << "\n"
            << Dune::isUnordered(10, 20)
            << "\n";

  return 0;
}
```

El comando `duneproject`

Es un asistente en el lenguaje `bash` que se encuentra en `/usr/bin/duneproject`…

Referencias

- Libros

  Oliver Sander. *DUNE — The Distributed and Unified Numerics Environment*. First. Lecture Notes in Computational Science and Engineering 140. Springer International Publishing, 2020. ISBN: 978-3-030-59701-6. DOI: 10.1007/978-3-319-03038-8.

- Artículos

  Peter Bastian y col. "The Dune framework: Basic concepts and recent developments". En: *Computers & Mathematics with Applications* 81.1 (1 de ene. de 2021). Development and Application of Open-source Software for Problems with Numerical PDEs, págs. 75-112. ISSN: 0898-1221. DOI: https://doi.org/10.1016/j.camwa.2020.06.007. URL: https://www.sciencedirect.com/science/article/pii/S089812212030256X.

- Sitios web

  Santiago Torres Arias, Jesús Castro y Andrea Gómez. *Taller de contribución a Arch Linux – Cumbre de Contribuidores de Open Source Software (CCOSS)*. 22 de oct. de 2020. URL: https://sg.com.mx/buzz/ponencias/ccoss-2020/taller-de-contribucion-arch-linux (visitado 10-03-2021).

  ArchWiki. *Arch Linux*. 27 de abr. de 2021. URL: https://wiki.archlinux.org/title/Arch_Linux (visitado 30-05-2021).