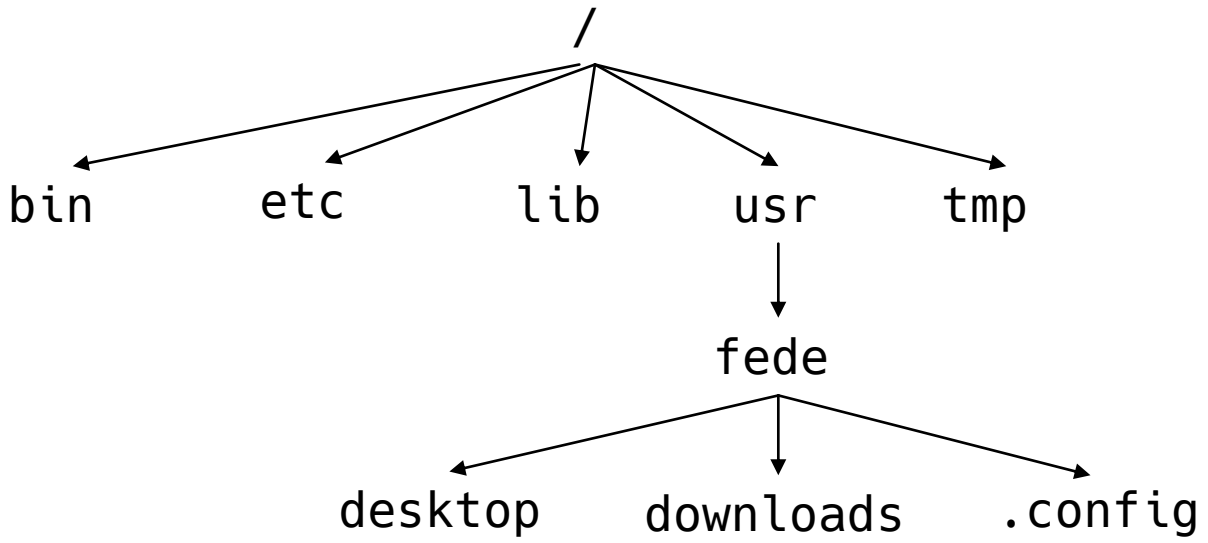
07 – Árboles Generales y TRIE

DOCENTE – FEDERICO VILENSKY

Recordemos

- Un árbol general con tipo base T es:
  - O bien, la estructura vacía
  - O bien, un nodo de tipo T, llamado raíz del árbol, junto con un número finito de estructuras de árbol, de tipo base T, disjuntas, llamadas subárboles
- Cuando decimos un numero finito, no es fijo, en cada nodo puede tener una cantidad distinta
- Cómo lo podríamos implementar?

Un ejemplo típico

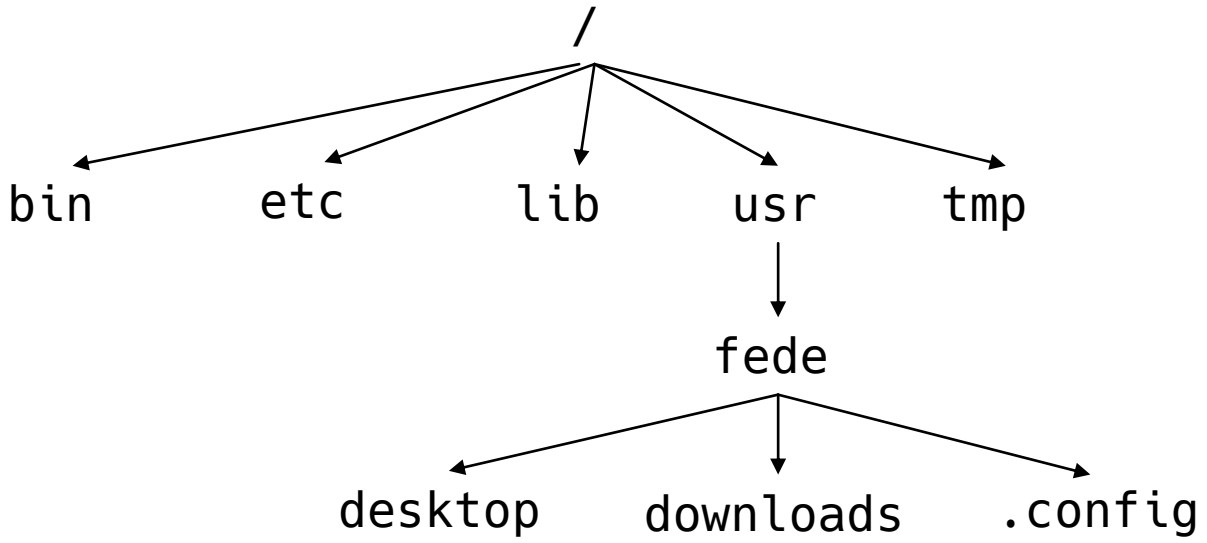


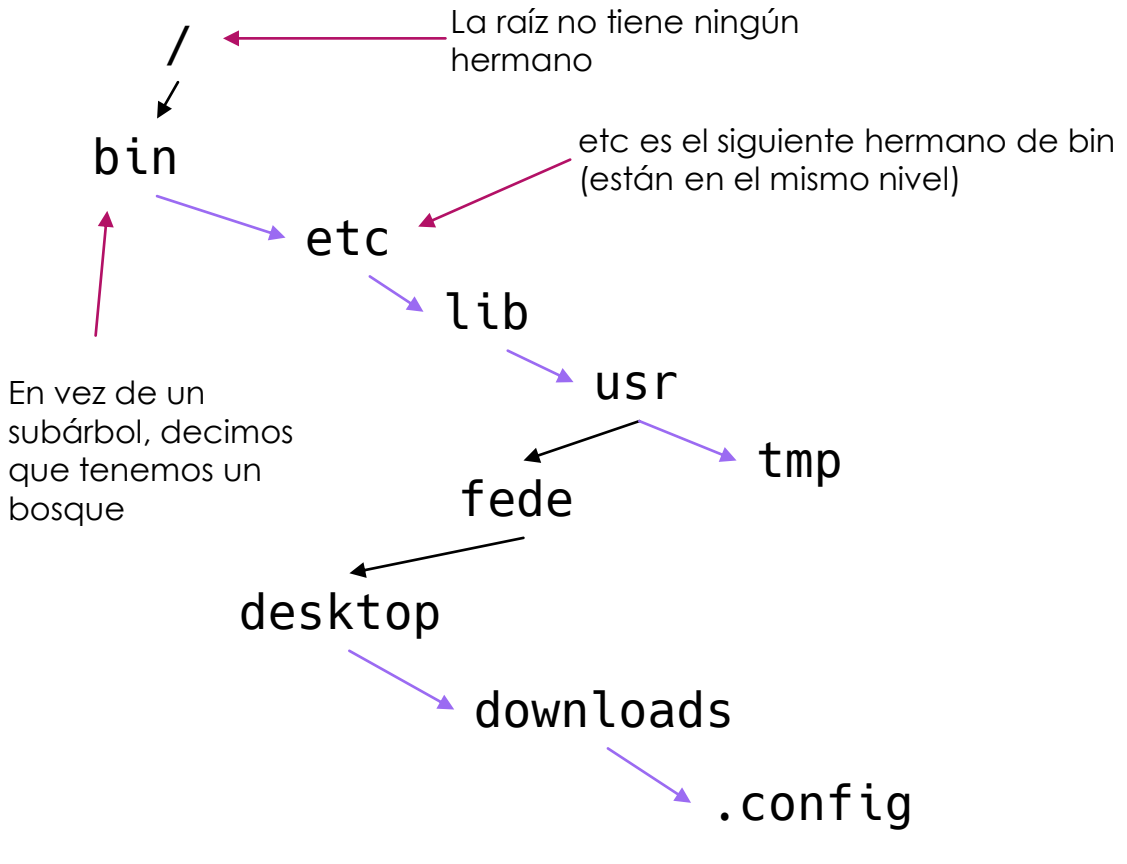
- El file system (estructura de las capretas) de un sistema operativo
- Sirve para, por ejemplo, listar las carpetas (recorrido de arboles)
  - Incluso hay una herramienta en la mayoría de los sistemas Unix, que se llama tree, que hace esto

Cómo lo representamos

- En un árbol general, el número de hijos por nodo puede ir cambiando.
- Entonces una primer idea es que cada nodo contiene una lista de subárboles
- Si seguimos pensando en como solucionarlo, otra forma es pensar que cada nodo va a tener a su **primer hijo** y a su **siguiente hermano**
  - **Primer hijo:** un nivel mas abajo
  - **Siguiente hermano:** mismo nivel
- Con esta noción, podemos implementarlos mediante el uso de arboles binarios, donde
  - Subárbol izquierdo, se corresponde con el primer hijo
  - Subárbol derecho, se corresponde con el siguiente hermano
- Ahora que sabemos como representar a los arboles generales con arboles binarios, podemos decir que ambos modelos son **equivalentes**
  - Con arboles binarios podemos representar a cualquier árbol general
  - Los arboles binarios son un caso particular de los arboles generales

Veamos el ejemplo anterior como AB






Veamos algunos
ejemplos con código

Contar nodos de un árbol general

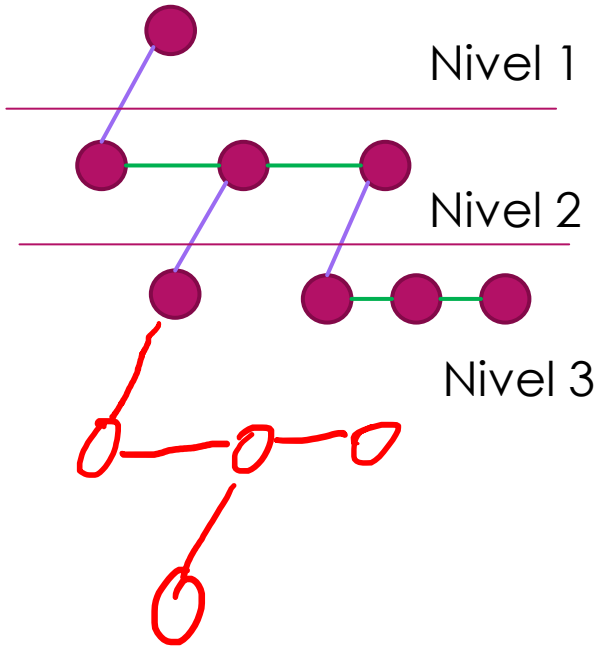
- Primero nos vamos a definir nuestro árbol general
- ```
  typedef NodoAG * AG;

  struct NodoAG
  {
    T item;
    AG pH;
    AG sH;
  };
  ```
- ```
  int contarNodos (AG t)
  {
    if (t == NULL) return 0;
    else return 1+ contarNodos (t->pH) + contarNodos(t->sH);
  }
  ```
- Este código es igual que al de árbol binario

Altura de un árbol general

```
int altura (AG t)
{
  if (t == NULL) return 0;
  else return max(1+altura(t->pH), altura(sH));
}
```

- OJO
  - No es el mismo código que el árbol binario
  - El primer hijo nos hace crecer la altura
  - El siguiente hermano NO crece la altura



Imprimir elementos del nivel k

- Queremos imprimir los elementos en el nivel k del árbol, asumiendo que la raíz de un árbol no vacío tiene nivel 1
- void imprimirNivel(AG t, int k)

```
{
  if(t != NULL && k > 0)
  {
    if(k == 1) {cout << t->item;}
    else {impNivel(t->pH, k-1);}
    impNivel(t->sH, k)
  }
}
```

Eliminar subárboles con raíz x

- Dado un árbol general, eliminar todos los subárboles que tengan a x como valor
-
```
void eliminarX(AG & t, T x){
  if(t != NULL) {
    if(t->item == x) {
      AG aux = t;
      t = t->sH;
      eliminarTodo(aux->pH);
      delete aux;
      eliminarX(t, x);
    }
    else{
      eliminarX(t->pH, x);
      eliminarX(t->sH, x);
    }
  }
}
```

Eliminar todo

```
void eliminarTodo(AG &t){
  if(t!=NULL){
    eliminarTodo(t->sH);
    eliminarTodo(t->pH);
    delete t;
  }
}
```

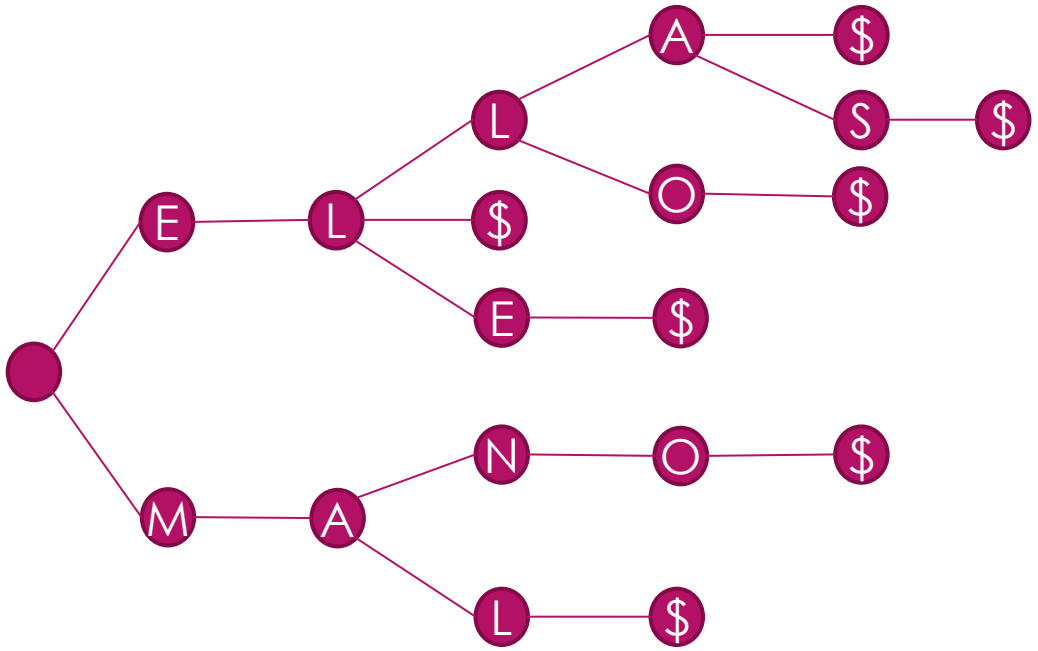
Dudas?

Árboles Trie

- Muchas palabras tienen prefijos en común
  - El, Ella, Elástico, Electricidad, Electrico
- El Trie, o árbol de prefijos, es una estructura que cuando tenemos muchas palabras con prefijos en común, en vez de tener que guardar el prefijo una vez para cada palabra, nos permite guardarlo una única vez
- Definición
  - Cada nodo (excepto la raíz) está etiquetado con un carácter (a,…,z) o con una marca de fin (símbolo $ o ε, dependiendo la bibliografía)
  - Un camino desde la raíz hasta una hoja (lo marcamos con $) es una palabra del diccionario
  - Cada nodo (salvo la raíz y las hojas) es un prefijo del conjunto

Ejemplo

- {EL, ELLA, ELLO, ELLAS, ELE, MANO, MAL}



Usos de un Trie

- Como ya mencionamos esta estructura sirve cuando tenemos muchas palabras que todas comparten prefijos particulares
  - Cuando tenemos una cantidad de prefijos << suma de las longitudes de las palabras
- Un nodo puede tener hasta 27 hijos (caracteres + $)
  - Generalmente mucho menos
- Se pueden también representar otros tipos de objetos
  - Enteros, reales, etc.
- Un caso de uso que aparece es en la teoría de autómatas, donde se utilizan como paso intermedio para construir autómatas
  - Sirven por ejemplo para sanitizers (cuando aceptamos texto en un input, que no nos puedan hacer una inyección SQL por ejemplo)
- Otro caso de uso es en bases de datos, por ejemplo las base de datos de las criptomonedas, se guardan en tries, ya que ahorran mucho espacio, y es vital que sean lo mas chicas posibles, ya que cada computadora debe descargarla para que funcione la red de cripto

FIN