1. ¿Qué dos procesos especiales existen en UNIX? ¿Qué jerarquía tienen los procesos en UNIX y quién está en el tope de dicha jerarquía?

Proceso init (PID: 1) y swap (PID: 0). Cada proceso es creado por otro proceso, el proceso que crea a otro se denomina padre. El proceso init crea al resto de procesos, por lo que es la raíz del árbol y está en el tope de la jerarquía.

2. Cita TODOS los identificadores de proceso que existen en UNIX e indica cuáles están relacionados con la seguridad del sistema.

- **Usuario propietario**: Usuario que ejecuta el proceso.
- **Usuario efectivo**: Usuario cuyos privilegios son los que tiene el proceso. (El proceso puede hacer todo lo que el usuario efectivo). // Implica un riesgo de seguridad.
- **Grupo propietario**: Grupo al que pertenece el usuario que ejecuta el proceso.
- **Grupo efectivo**: Grupo cuyos privilegios son los que tiene el proceso. (El proceso puede hacer todo lo que el grupo efectivo). // Implica un riesgo de seguridad.

3. Implementa el comando UNIX idproc que imprima en pantalla (con mensajes completos y claros) todos los identificadores del proceso.

Código (C++):
```
#include <iostream>
#include <unistd.h>
int main ()
{
	std::cout << “Usuario propietario: “ << getuid()  << std::endl;
	std::cout << “Usuario efectivo: “    << geteuid() << std::endl;
	std::cout << “Grupo propietario: “   << getgid()  << std::endl;
	std::cout << “Grupo efectivo:        << getegid() << std::endl;
}
```

4. ¿Cuándo tiene un proceso máximos privilegios? Implementa un comando UNIX MaxPriv que cambie los privilegios de un proceso a privilegios de superusuario (máximos privilegios) y, posteriormente, vuelca a la pantalla información que verifique que se han obtenido tales privilegios.

Un proceso tiene máximos privilegios cuando el usuario efectivo root (UID: 0).

Código (C++):
```
#include <iostream>
#include <unistd.h>
int main ()
{

	std::cout << “Usuario propietario: “ << getuid()   << std::endl;
	std::cout << “Usuario efectivo: “    << geteuid()  << std::endl;
	setuid((uid_t) 0);
	std::cout << “setuid((uid_t) 0);”                  << std::endl;
	std::cout << “Usuario propietario: “ << getuid()   << std::endl;
	std::cout << “Usuario efectivo: “    << gete uid() << std::endl;
}
```

Salida:
```
UO281827@gollum:~/SO/E2/2/maxPriv$ ./maxPriv
Usuario propietario: 2090
Usuario efectivo: 2090
setuid((uid_t) 0);
Usuario propietario: 2090
Usuario efectivo: 2090
UO281827@gollum:~/SO/E2/2/maxPriv$
```

El programa no retorna excepción ni error, pero tampoco cambia el usuario efectivo a root.

5. ¿Qué tres tipos de acceso (permisos) hay en UNIX? Explica cómo funcionan los tres tipos de permisos UNIX en ficheros y en directorios. ¿Qué hacen los comandos UNIX: chown, chgrp y chmod?

Lectura (r)
Escritura (w)
Ejecución (x)
Se permite seleccionar cualquier combinación de los 3 permisos para el usuario actual, para el grupo al que pertenece el usuario y para otros. Por lo que finalmente hay 9 permisos, (r w x) repetidos 3 veces para cada 'grupo'.

Chown: Permite cambiar el usuario y grupo propietario de ficheros.
Chgrp: Permite cambiar el grupo propietario de ficheros.
Chmod: Permite cambiar permisos concretos a un 'grupo' concreto.

6. Implementa un comando printenv que vuelque a la salida todo el entorno del proceso.

Código (C++):

```
#include <iostream>
#include <stdlib.h>
int main (int argc, char* argv[], char* envp[])
{
	for (int i = 0 ; envp[i] ¡=NULL ; i++) {
		std::cout << envp[i] << std::endl;
	}
}
```