Sesión 2:

- **condicionales y test**
- **select case**
- **for**

AC1: La sentencia if_then_fi, if_then_else_fi y el test

- Lee individualmente el apartado 5.2.1 de los apuntes. 5'
- Repartiros los tres tipos de condiciones que se pueden hacer con la instrucción ***test*** entre los miembros del grupo. Para ello podéis utilizar los siguientes ejemplos:

```
# Ejemplo 1 (numérico)
usuarios=`who | wc -l`
if test "$usuarios" -gt 8
then
        echo mas de 8 personas conectadas
else
        echo menos de 8 personas conectadas
fi
```

```
# Ejemplo 2 (sobre ficheros)
if test ! -d "$1"
then
        echo directorio no valido
else
        echo directorio valido
fi
```

```
#Ejemplo 3 (sobre cadenas de caracteres)
if test -z "$1"
then
        echo introduzca un argumento
else
        echo el argumento es $1
fi
```

- Reuniros por grupos de expertos. Estudiar qué hace cada ejemplo. 10 minutos.
- Volver al grupo original y compartir con los demás lo que hemos visto. Aclarar las dudas. 15'.

AC2: La sentencia case_esac

- Leer individualmente el apartado 5.3 de los apuntes. 5’
- Sin cambiar de grupo, estudiar detenidamente el siguiente ejemplo. 10’

```
# Ejemplo 4: manejo de la instrucción case_esac
# recibe 2 argumentos: un carácter y el nombre de un fichero en el directorio actual
case "$1" in
l|L)
  echo "el numero de lineas en $2 es " `wc -l $2`;;
p|P)
  echo "el numero de palabras en $2 es " `wc -w $2`;;
c|C)
  echo "el numero de caracteres en $2 es " `wc -c $2`;;
*)
  echo "primer argumento desconocido" ;;
esac
```

- Una vez que hayas comprendido el ejercicio anterior, propón un nuevo guión o programa del shell que acepte exactamente 4 argumentos. Si el número de argumentos es menor de 4, deberá dar un mensaje de “Numero de argumentos insuficiente”, si el número de argumentos es mayor que 4 deberá dar un mensaje similar “Numero de argumentos excesivo”. Si el número de argumentos es correcto, también deberá indicarlo. Para facilitar el análisis de los casos utiliza la sentencia *case_esac*. 15’.

AC3: La sentencia for_done

- Leed individualmente el apartado 5.4 de los apuntes. 5'
- Repartiros por grupos de expertos. Cada uno debe estudiar cómo funciona cada una de las tres variantes del mandato *for_done*. 10'.
  Podéis utilizar los siguientes programas de ejemplo:

```
# Ejemplo 5: el esquema for_do_done, recorriendo una lista de
# argumentos del guion/programa shell
for var in $1 $2 $3
do
        echo "el argumento es $var"
done

## Ejemplo 6
for var in $*                   ## es equivalente a poner: for var
do
        echo "el argumento es $var"
done
```

```
# Ejemplo 7: El esquema for_do_done, recorriendo una lista que se genera
# a partir de un mandato
echo uno dos tres cuatro cinco > arch.txt
for var in `cat arch.txt`
do
        echo el argumento es $var
done
```

```
# Ejemplo 8:  El esquema for_do_done, recorriendo la lista generada
# a partir de una  expresión regular
for var in *
do
        echo "El fichero es $var"
done

## Ejemplo 9
for var in $HOME/bin/*
do
        echo "El fichero es $var"
done
```

- Volved al grupo original y comentad con vuestros compañeros lo que habéis observado, explicando el funcionamiento distinto de cada ejemplo. 15'. Intentad resolver el siguiente enunciado:

Ejercicio: Usando todos los conceptos que has aprendido hoy, haz un programa del shell que muestre los nombres de todos los directorios que haya en tu directorio HOME.