TOTALE 09/09/2019

1. Indicare quale dei seguenti comandi consente di modificare i permessi del file "pippo" dalla maschera r-- -w- r-x alla maschera rw- r-- --- : chmod 462 pippo; chmod o-w pippo
2. Sia – r—rwx r-x la maschera di accesso di un nodo del file system, quale delle seguenti affermazioni è VERA? : il proprietario ha i diritti di lettura
3. Il comando file /usr/bin/file stampa : il tipo del file "/usr/bin/file"
4. Il comando egrep "pippo|pluto|paperino" pippo.txt pluto.txt paperino.txt stampa : Le linee dei tre file di testo che contengono la stringa "pippo" oppure "pluto" oppure "paperino"
5. All'interno di uno script, la variabile $@ contiene : Tutti gli argomenti passati allo script
6. Il comando su sta per : Substitute User
7. Il comando sed s/$/'?'/g sed.cmd : Stampa una copia del contenuto del file "sed.cmd" dopo aver appeso il carattere "?" alla fine di ogni linea
8. Per redirigere sia lo Standard Output che lo Standard Error del comando "ls" nel file "output.log" dovrei eseguire : ls > output.log 2>&1
9. Il comando "ls -la | egrep "^-" | tail -n 1" stampa : Le informazioni relative all'ultimo contenuto di tipo file nella lista dei contenuti stampata dal comando "ls"
10. La sequenza di comandi var1='Ciao' && var2='$var1 a tutti' && echo $var2 stampa : -bash: Ciao: command not found
11. Scrivere nelle tre righe sotto cosa stampa il seguente script:

```
#!/bin/bash

VAR1=6

function test_var (){
  export VAR2=8
  local VAR 3=$(($VAR1-$VAR2))
  export VAR3
}

test_var
echo $VAR1
echo $VAR2
echo $VAR3
```

_____6_____

_____8_____

_____

12. Completare il seguente script che, per ogni file nella directory "./ppp" con estensione ".apt", il cui nome inizia con "a" oppure "k" e finisce con "!*", stampa il numero di stringhe "pippo" contenute nel file se questo numero è maggiore o uguale a 4, "Not enough!" altrimenti.

```
#!/bin/bash
for file in  ./ppp/[ak]* '!' '*' .apt
do
   count=$( egrep -c pippo $file )
   if test $count -ge 4
   then
        echo "$file:  $count"
    else
        echo "$file: Not enough!"
    fi
    done
```

13. Usare comandi di shell bash da terminale per stampare in output tutte le informazioni delle directory entry della directory corrente, una riga per ogni entry, avendo cura di indicare tutte le entry di tipo directory con la "D" al posto di "d". Usare esclusivamente la riga sotto.
    _____ls -l | sed s/^[d]/D/g_____
    _____ls -l | tr d D_____

14. Completare il seguente programma C che legge dallo standard input delle parole e ne stampa la loro concatenazione nello standard output.

```
#include …
  char str[200], strconcat[4096];
printf("Inserisci sequenza di parole separate da [INVIO]. Terminare con [CTRL-D] \n");
while(scanf("%s", str) != EOF) {
        strcat(strconcat, str);
}
printf("La concatenazione è: %s\n", strconcat);
}
```

15. Utilizzando esclusivamente comandi di shell bash, scrivere nella riga sotto cosa fa il seguente programma:
    cmd arg > file (per esempio ./es16 pippo.txt ls - la esegue ls - la > pippo.txt)

```
#include …

int main (int argc, char * argv[])
{
      int fd;
      if (argc < 3 ){
          printf("Usage: esercizio15 file cmd arg …\n");
          return (0);
      }
      if ((fd = open(arg[1], O_WRONLY|O_CREAT|O_TRUNC, 0644)) == -1) {
          printf("Error: opening file %s \n", argv [1]);
          return (-1);
      }
      dup2(fd, 1);
      close(fd);
      execvp(argv[2], &argv[2]);
      printf("Error on %s\n", argv [2]);
      return(-1);
}
```

16. Utilizzando esclusivamente comandi di shell bash, scrivere nella riga sotto cosa fa il seguente programma:
    ___du -a | sort -nr___

```
#include …

void gogo();
```

```
int main (int argc, char **argv) {
        int pid, status;
        int fd[2];

        pipe(fd);
        switch (pid = fork()) {
        case 0:
                gogo (fd);
                exit(0);
        default:
                while ((pid = wait(&status)) != -1)
                        fprintf(stderr, "process %d exits with %d\n", pid, WEXITSTATUS(status));
                break;
        case -1:
                perror("fork");
                exit(1);
        ;
        exit(0);
}
char*pippo[] = { "sort", "-nr", 0 };
char *pluto[] = { "du", "-a", 0 };
void gogo(int pid()) {
        int pid;
        switch (pid = fork()) {
        case 0:
                dup2(pfd[0], 0);
                close(pfd[1]);
                execvp(pippo[0], pippo);
                perror(pippo[0]);
        default:
                dup2(pfd[1], 1);
                close(pfd[0]);
                execvp(pluto[0], pluto);
                perror(pluto[0];
        case -1:
                perror("fork");
                exit(1);
        }
}
```