# Tasks for the class test

## Task One Revision

1. Revise week 1 Introduction to operating systems slide 1-11
2. Revise week 2 Security and business and virtual box

## Task Two

Research and list 5 benefits and 5 risks for using Linux in a commercial environment, adding a source for each one

|  |  |
| --- | --- |
| Benefits | Sources |
| Various distros are available; there are an abundance of distros targeted to different industries. Users can use a distro that’s optimal for their work type e.g., a particular distro targeted for cybersecurity will come pre-installed with hacking and networking software already pre-installed. | Ankush Das (09/2020). *11 reasons why linux is better than windows.* Retrieved from: https://itsfoss.com/linux-better-than-windows/.[accessed on 23/01/2022] |
| Security: Linux is divided into so many distros that barely any viruses and hacks have been created for them, therefore linux tends to be more secure compared to other more popular OS. | Kumar, D., Singh, M. and Bargotra, R., (2019). *OPERATING SYSTEMS: PROS & CONS.* [accessed on 23/01/2022] |
| Personal: With so many distros available there are an abundance of different UIs which are more simple or complex and cater to a wider range of tastes | Trappler, T., (2009). *Is there such a thing as free software? The pros and cons of open source software*. EDUCAUSE quarterly, *32*(2), p.10..[accessed on 23/01/2022] |
| Virtualization: Linux is often utilised for the fact that it can be easily opened in a virtual machine, meaning that any configuration done on it has no lasting effect on the user’s actual computer. Therefore, people can test their ideas and work on a separate platform within their computer. | Kolyshkin, K., (2006). *Virtualization in linux*. White paper*,* OpenVZ, *3*(39), p.8. [accessed on 23/01/2022] |
| Open source: linux is open-source, and all distors are created and managed by a committed community. This means there’s transparency and users can have faith in the software they’re using. Additionally, users can trust that they’re not going to become products to a company. | DiBona, C., Ockman, S. (2008). *Open Sources: Voices from the Open Source Revolution*. United States: O'Reilly Media. .[accessed on 23/01/2022] |

|  |  |
| --- | --- |
| Risks | Sources |
| Community driven: The fact that it’s community driven means that patches and updates are released at an individual’s leisure/free time, as opposed to a company being driven to create a fix. So if issues or breaches were to occur, there’s no guarantee of a quick patch. | Bob Toxen. (2007). *The Seven Deadly Sins of Linux Security: Avoid these common security risks like the devil*. Pg 38–47. DOI:https://doi.org/10.1145/1255421.1255423. [accessed on 23/01/2022] |
| Forgotten Distros: Because distros for linux are created and maintained by a community, some distros can be completely forsaken. Therefore, no updates are introduced and a lot of distros will fail to compete with newer and updated OS’s. | Jim Lynch, (02/2014). *The magic of disappearing linux distros*. Retrieved from: https://www.infoworld.com/article/2831572/the-magic-of-the-disappearing-linux-distros.html . [accessed on 23/01/2022] |
| Unfriendly UI: Although some UI’s will be preferred by users, some UI’s will be hard for users to manoeuvre with. Distros are created by other individuals as opposed to a company so it’ll lack finesse and polishing that other OS’s can provide. | Dantin, U., (01/2005). *Application of personas in user interface design for educational software*. In *Proceedings of the 7th Australasian conference on Computing education-Volume 42* (pp. 239-247). [accessed on 23/01/2022] |
|  | [accessed on 23/01/2022] |
|  | [accessed on 23/01/2022] |

# Task 3: Linux commands

Using the lectures and tutorials for week 2-3 as a guide, complete the following questions using Terminal in your Linux Mint VM, adding the commands you enter or screen shots you take in the right-hand column.

When typing commands, Microsoft Word may convert some punctuation including hyphens and double quotes into “smart punctuation” like dashes and curly quotes. These will not work if you attempt to copy them back out into a VM. You can [disable the smart quotes feature](https://support.microsoft.com/en-us/office/smart-quotes-in-word-702fc92e-b723-4e3d-b2cc-71dedaf2f343) in Word and turn off “Hyphens with dash” in the same place.

Where we have stated *[your\_initials]* you must replace this with **your own initials** i.e. for *[your\_initials]-question2*, Fiona would put *FK-question2*

|  |  |
| --- | --- |
| Question | Command Entered |
| 1. Go to your home directory | Cd |
| 1. Output a simple list of the contents of the home directory | Ls -l |
| 1. Create subdirectory called *[your\_initials]-dir* | mkdir SPDS-dir |
| 1. Change directory into *[your\_initials]-dir* | Cd SPDS-dir |
| 1. Using one command, create and edit a file called *[your\_initials]-test.txt*, adding the following lines of text   EPERM 1 Operation not permitted  ENOENT 2 No such file or directory  ESRCH 3 No such process  EINTR 4 Interrupted system call  EIO 5 Input/output error  ENXIO 6 No such device or address  E2BIG 7 Argument list too long  NB: you only need to show the command to create and edit the file, not how you put the data in the file | Nano SPDS-test.txt  EPERM 1 Operation not permitted  ENOENT 2 No such file or directory  ESRCH 3 No such process  EINTR 4 Interrupted system call  EIO 5 Input/output error  ENXIO 6 No such device or address  E2BIG 7 Argument list too long |
| 1. Output the contents of *[your\_initials]-test.txt* | Cat SPDS-test.txt |
| 1. Take a screen shot of the output of   *[your\_initials]-test.txt* |  |
| 1. Use a command to output all the lines in   *[your\_initials]-test.txt* that contain the word ‘No’ | Grep “No” SPDS-test.txt |
| 1. Use numeric notation to set the permissions for *[your\_initials]-test.txt* to give the owner, group, and other users read access, but not write or execute | Chmod 444 |
| 1. Output the directory contents of *[your\_initials]-dir* with permissions and sizes | Ls -l SPDS-test.txt |
| 1. Take a screen shot of the contents of *[your\_initials]-dir* with permissions and sizes |  |
| 1. Create a sub-directory called *[your\_initials]-favourites* | Mkdir SPDS-favourites |
| 1. Change directory into *[your\_initials]-favourites* | Cd SPDS-favourites |
| 1. Use shell redirection to output your favourite colour into *[your\_initials]-colour.txt* | echo “red” > SPDS-colour.txt |
| 1. Write your favourite food into *[your\_initials]-food.txt* | Echo “lasagna” > SPDS-food.txt |
| 1. Create a file called *anon.txt* | Touch anon.txt |
| 1. Output the contents of *[your\_initials]-favourites* with permissions and sizes | Ls -l SPDS-favourites.txt |
| 1. Take a screen shot of the contents of   *[your\_initials]-favourites* with permissions and sizes |  |
| 1. Output only those files that start with *[your\_initials]* showing the permissions and sizes | Ls -l SPDS-\*.txt |
| 1. Take a screen shot of those files that start with *[your\_initials]* showing the permissions and sizes |  |
| 1. Delete *anon.txt* | Rm anon.txt |
| 1. Change directory to the parent directory *[your\_initials]-dir* | Cd SPDS-dir |
| 1. Without changing to *[your\_initials]-favourites*, output the contents of *[your\_initials]-favourites* with permissions and sizes | Ls -l SPDS -favourites |
| 1. Take a screen shot of the contents of   *[your\_initials]-favourites* with permissions and sizes |  |
| 1. Withoutchanging to *[your\_initials]-favourites*, output the contents of *[your\_initials]-food.txt* | Cat SPDS-favourites/SPDS-food.txt |
| 1. Withoutchanging to *[your\_initials]-favourites*, delete *[your\_initials]-food.txt* | Rm SPDS-favourites/SPDS-food.txt |
| 1. Delete *[your\_initials]-favourites* subdirectory | Rm -r SPDS-favourites |
| 1. Find all files in the */usr* directory and subdirectories called *README.md* | Find /usr -name README.md |
| 1. Find all .jpg files in the */usr/share directory* and subdirectories | Find /usr /share \*.jpg |
| 1. Add a user called *[your\_initials]user*, creating their home directory immediately, setting their shell as */bin/bash* and giving them a comment of *[your\_initials]* | Sudo useradd -s /bin/bash -c “SPDS” SPDSuser -m |
| 1. Take a screen shot showing the home directory for all users |  |
| 1. Use a terminal command to search the */etc/passwd* file for the new user's username | Grep “SPDSuser” /etc/passwd |
| 1. Change the new user's password to   *[your-initials]-banana* | Sudo passwd SPDSuser  SPDS-banana |
| 1. Change directory to the system temporary directory at */tmp* | Cd /tmp |
| 1. Create a subdirectory called *[your-initials]-share* | Mkdir SPDS-share |
| 1. Set the permissions of *[your-initials]-share* so that everyone has full access | Chmod 777 SPDS-share |
| 1. Take a screen shot of the contents of   *[your\_initials]-share* with permissions and sizes |  |
| 1. Change the owner of *[your-initials]-share* to *[your-initials]user* | Sudo chown SPDSuser SPDS-share |
| 1. Change directory into *[your-initials]-share* | Cd SPDS-share |
| 1. Switch user to *[your-initials]user* | Su SPDSuser |
| 1. Use shell redirection to write the current username into a file called *[your-initials]-username.txt* (do not type your name as part of this command) | Whoami>SPDS-username.txt |
| 1. Take a screen shot of the contents of *[your-initials]-username.txt* |  |
| 1. Use a terminal command to end your switch user session | Exit |
| 1. Take a screen shot showing *[your-initials]-username.txt* is owned by *[your-initials]user* |  |
| 1. Delete *[your-initials]user* and their home directory | Sudo userdel -r SPDSuser |

# Task 4: Bash script

Complete the following, adding the Bash script, the commands you entered or screen shots you took in the right-hand column.

You **MUST NOT** include your code as a screenshot. We require you to provide the text in case we need to test them. Retype your code carefully, or use the shared folder method to access the scripts you write in your host operating system to copy and paste

Where we have stated *[your\_initials]* you must replace this with **your own initials** i.e. for *[your\_initials]-question2*, Fiona would put *FK-question2*

|  |  |
| --- | --- |
| Question | Bash script source code / terminal command(s) entered to run the script / screenshot showing output of running the script |
| 1. Create a file called *[your-initials]errors.txt* containing the following text   1 EPERM Operation not permitted  2 ENOENT No such file or directory  3 ESRCH No such process  4 EINTR Interrupted system call  5 EIO I/O error  6 ENXIO No such device or address  7 E2BIG Argument list too long  8 ENOEXEC Exec format error  9 EBADF Bad file number  10 ECHILD No child processes  11 EAGAIN Try again  12 ENOMEM Out of memory  13 EACCES Permission denied  14 EFAULT Bad address  15 ENOTBLK Block device required | Nano SPDSerrors.txt  1 EPERM Operation not permitted  2 ENOENT No such file or directory  3 ESRCH No such process  4 EINTR Interrupted system call  5 EIO I/O error  6 ENXIO No such device or address  7 E2BIG Argument list too long  8 ENOEXEC Exec format error  9 EBADF Bad file number  10 ECHILD No child processes  11 EAGAIN Try again  12 ENOMEM Out of memory  13 EACCES Permission denied  14 EFAULT Bad address  15 ENOTBLK Block device required |
| 1. Write a bash script called *bash1.sh* to  * prompt for and read your *[your-initials]* * output *Hello [your-initials]* * prompt for and read a number between 1 and 5 and store it in a variable *<number>* * output the last *<number>* of lines of the *[your-initials]errors.txt* without using a loop | Nano bash1.sh  #!/bin/bash  Echo “Please enter your name: ”  Read name  Echo “Hello $name”  Echo “Please enter a number between 1 and 5: ”  Read number  Tail -$number SPDSerrors.txt |
| 1. Enter the command to change permissions so that *bash1.sh* can be executed | Chmod u+x bash1.sh |
| 1. Enter the command to run *bash1.sh* - when your script is running and prompts you for a number, enter 5 | ./bash1.sh  Salvatore  5 |
| 1. Take a screen shot showing all output from this script |  |
| 1. Write a bash script called *bash2.sh* to:  * take a username as a parameter (do not use read in the script) * append that username and the current date and time to a file called *[your-initials]-out.txt* | Nano bash2.sh  #!/bin/bash  Username=$1  Date\_and\_time=$(date+%H:%M)  Echo “$username + $date\_and\_time”>>SPDS-out.txt |
| 1. Run the *bash2.sh* three times using the following usernames:  * *[your-initials]-1* * *[your-initials]-2* * *[your-initials]-3* | Chmod u+x bash2.sh  ./bash2.sh SPDS-1  ./bash2.sh SPDS-2  ./bash2.sh SPDS-3 |
| 1. Take a screen shot showing the contents of   *[your-initials]-out.txt* |  |
| 1. Write a bash script called *bash3.sh* to:  * take a domain to download and a word to search for as parameters (do not use read in the script)   download the URL given as the first parameter and search the output for the phrase provided as the second parameter  N.B. You should pipe the output of one command to the input of another to accomplish this. | Nano bash3.sh  #!/bin/bash  Wget -q $1 -o - | grep -colour “$2” |
| 1. Take a screen shot showing the output when searching *https://www.staffs.ac.uk* for the word *Computer* |  |
| 1. Write a bash script called *bash4.sh* to:  * prompt for and read a nickname * add the nickname to a file called *[your-initials]-name.txt* replacing any existing contents of the file * output whether or not the current user’s username and the nickname in *[your-initials]-name.txt* are the same | nano bash4.sh  #!/bin/bash  echo “Please enter your nickname: “  read nickname  echo $nickname>SPDS-name.xtx  user=$(whoami)  if [[ $nickname == $user ]]  then  echo “Both the User and the Nickname are the same”  else  echo “Both the User and the Nickname are not the same”  fi |
| 1. Take two screen shots, showing the output where:  * names are the same * names are different |  |
| 1. Create four files in a new directory called *[your-initials]-5*:  * *[your-initials]-1.sh* * *[your-initials]-2.sh* * *[your-initials]-3.sh* * *[your-initials]-4.txt* * add *#!/bin/bash* to the start of *[your-initials]-1.sh* * add *[your-initials]-WDOS* to the start of *[your-initials]-2.sh* * do not add any content to *[your-initials]-3.sh* or *[your-initials]-4.txt* | touch SPDS-{1..3}.sh  touch SPDS-4.txt  nano SPDS-1.sh  #!/bin/bash  nano SPDS-2.sh  SPDS-WDOS  chmod a=rw SPDS-{1..3}.sh  chmod a=rw SPDS-4. |
| 1. Take a screen shot of the directory content showing the permissions of the four files and illustrating that neither the user, group, nor everyone has execute access |  |
| 1. Write a bash script called *bash5.sh* to:  * change the permissions of every *.sh* file in the directory passed as a parameter so it may be executed by the user * output *[your-initials]* followed by the number of *.sh* files * output one of the following messages as appropriate for each .sh file:   + "file is empty"   + "file contains #!/bin/bash"   + "file missing #!/bin/bash" | !/bin/bash  counter=0  for file in $@  do  if echo "$file" | grep -q ".sh"  then  chmod u+x $file  ((counter++))  fi  done  echo "SPDS has ( $counter ) .sh files"  for file in $@  do  if grep -q "#!/bin/bash" $file  then  echo "The $file File contains #!/bin/bash"  elif [[ -s "$file" ]]  then  echo "The $file File is missing #!/bin/bash"  else  echo "The $file File is empty"  fi  done |
| 1. Take a screen shot showing the permissions of the four files and the output of the appropriate messages |  |
| 1. Write a bash script called *bash6.sh* to:  * prompt for and read an email address, e.g. [a000001a@student.staffs.ac.uk](mailto:a000001a@student.staffs.ac.uk) * use a function called email\_to\_url to:   + extract the student username from the email address, e.g. a000001a   + output the web hosting URL by appending the username to the string <http://web.fces.staffs.ac.uk/> | nano bash6.sh  #!/bin/bash  echo "Please Enter your email: "  read email  email\_to\_url(){  username=$(echo $email | cut -d@ -f1 )  url="http://web.fces.staffs.ac.uk/$username"  }  email\_to\_url  echo "The username is : $username"  echo "The web Hosting URL is: $url" |
| 1. Run the script using your own university email address | **No output required for this question** |
| 1. Take a screen shot showing the web hosting URL output |  |
| 1. Write a bash script called *bash7.sh* to:  * use a loop to prompt for and read a word until a 5-letter word is entered * output "*[your-initials] entered the word*" followed by the word and a count of the number of words entered within the loop * prompt for and input a name and save it in a variable called *myname* * use a loop to output each character of *myname* on a separate line | #!/bin/bash  echo "Please enter a Word: "  read word  counter=(0)  until [[ ${#word} -eq 5 ]]  do  ((counter ++))  echo "Please enter a Word: "  read word  done  echo "SPDS has entered the word $word in $counter loops"  echo "Please input a name: "  read myname  grep -o . <<< $myname |
| 1. Take two screen shots, where:  * 5 letter word entered when first prompted * 5 letter word not entered when first prompted |  |
| 1. Enter the command to re-run *bash7.sh*, redirecting all output to a file called *[your-initials]-output.log*   NB: when you are testing this, all output will be sent to the log so you will not see any prompts - remember the order your script asks for input so you can provide it even when you don't see the prompt in your terminal screen. | ./bash7.sh > SPDS-output.log |
| 1. Write a bash script called *bash8.sh* with the exactly the following code, including the errors   #!/bin/bash  bug="cricket"  echo "$bug"  echos "my favourite bug is ${bug}"  bug = "ladybird"  echo "${bug:4} | **No output required for this question** |
| 1. Take a screen shot of the output |  |
| 1. Run the script again in script debug mode for the whole file | bash –x bash8.sh |
| 1. Take a screen shot of the output |  |
| 1. Add set commands to turn debug on at the beginning and turn debug off after the line beginning with *echos* | **#!/bin/bash**  **Set -x**  **bug="cricket"**  **echo "$bug"**  **echos "my favourite bug is ${bug}"**  **set +x**  **bug = "ladybird"**  **echo "${bug:4}** |
| 1. Run the script, redirecting *stdout* to a file called *[your\_initials]-output.log*, and *stderr* to a file called *[your\_initials]-error.log* | ./bash8.sh 2>SPDS-error.log 1>SPDS-output.log |
| 1. Take a screen shot of the contents of *[your\_initials]-output.log* |  |
| 1. Take a screen shot of the contents of *[your\_initials]-error.log* |  |
| 1. Correct the errors in *bash8.sh* | **#!/bin/bash**  **bug="cricket"**  **echo "$bug"**  **echo "my favourite bug is ${bug}"**  **bug="ladybird"**  **echo "${bug:4}”** |
| 1. Run the file again and take a screen shot |  |
| 1. Write a bash script called *bash9.sh* to:  * create an array of 5 names, including your forename, called *names* * output the contents of *names* * create a temporary directory (*tempdir)* named /tmp/*[your\_initials]-* followed by 7 randomly selected characters (eg. /*tmp/FK-g7k£hla*) * output the name of *tempdir* * for each *name* in *names*   + create a file called *name*. followed by 6 random characters e.g. *fiona.g6Czla*   + add the *name* to the file followed by " *is very clever*" e.g. *fiona.g6Czla* would contain "*fiona is very clever*"   + output the content of the file * list full details of the files in *tempdir* | nano bash9.sh  #!/bin/bash  names=( Salvatore Leon Jordan Eryk Matt)  echo "${names[\*]}"  tempdir=$(mktemp -d /tmp/SPDS-m2£2arb.XXXXXXXX)  echo "$tempdir"  for name in "${names[@]}"  do  tempfile="$(mktemp $tempdir/$name.XXXXXX)"  echo "The $name is very clever">$tempfile  cat $tempfile  done  ls $tempdir |
| 1. Run the code and take a screen shot of the output |  |
| 1. QUESTION BLANK | **No output required for this question** |
| 1. Write a bash script called *bash10.sh* to:  * generate a random number between 0 and 20 * output this number for testing purposes * uses an until loop to prompt for and read a guess until the guess equals the random number   + if the guess is correct, stop the loop and output a congratulations message and the number of guesses taken   + if the guess is incorrect, output either “your guess is too high” or “your guess was too low” as appropriate | #!/bin/bash  guess=0  retVal=$(( $RANDOM % (21) ))  echo "$retVal"  until [[ $num -eq $retVal ]]  do  guess=$(($guess+1))  echo "Guess the random number (0 to 20); "  read num  if [[ $num -lt $retVal ]]  then  echo "Guess to low"  else  echo "Gues to high"  fi  done  echo "Congratulations!"  echo "This is how many times you took a guess: $guess" |
| 1. Take two screen shots, where:  * correct guess first time * 3 guesses with at least one guess too high and one too low |  |
| 1. Write a bash script called *bash11.sh* to:  * store an array with an ordered list of 4 questions * store a second array with an ordered list of answers to the questions * output each question and prompt for an answer until the answer is correct * output the number of guesses take to complete the quiz | #!/bin/bash  Array1=( "1+1=?" "2+2=?" "3+3=?" "4+4=?" )  array2=( 2 4 6 8 )  item=0  answer=0  counter=0  while [ $item -lt ${#array1[@]} ]  do  echo "${array1[$item]}"  echo "Please enter your answer: "  read answer  if [ $answer -eq ${array2[$item]} ]  then  echo "Correct!"  item=$(( $item + 1 ))  ((counter++))  else  echo "Incorrect!"  ((counter++))  fi  done  echo "You managed to completed the quiz in $counter attempts" |
| 1. Take a screen shot showing at least one question being incorrect the first time it is answered |  |
| 1. Write a bash script called *bash12.sh* to:  * create a dictionary containing 3 entries with keys:   + *[your\_initials]-1*   + *[your\_initials]-2*   + *[your\_initials]-3* * and a values containing a password reminder clue * prompt for and read a key until the key stop is input   + if the key is in the dictionary, output the password reminder clue   + if the key is not in the dictionary, prompt to ask whether to store the new key in the dictionary   + if yes, prompt and read the password reminder clue and add it to the dictionary with the key * at the end of processing, output all key-value pairs in the dictionary to *[your\_initials]-reminder.txt* | #!/bin/bash  input=0  end="stop"  declare -A d  d[SPDS-1]="Clue1"  d[SPDS -2]="Clue2"  d[SPDS -3]="Clue3"  while [[ $input != $end ]]  do  echo "Please enter an existing key: "  read input  if [ -v d[$input] ]  then  echo "${d[$input]}"  elif [[ ! -v d[$input ]] && [[ $input != $end ]]  then  echo "Would you like to add new key in dictionary?"  read newdic  if [ $newdic = "yes" ]  then  echo "Please enter the password Clue: "  read passwordclue  echo "Please enter key: "  read key  d[$key]="$passwordclue"  fi  fi  done  for key in "${!d[@]}"  do  echo " $key ${d[$key]}">> SPDS -reminder.txt  done |
| 1. Take four screen shots:  * correct key entered * incorrect key entered, and new entry added to dictionary * incorrect key entered, but new entry not added to dictionary * contents of *[your\_initials]-reminder.txt* |  |
| 1. Write a bash script called *bash13.sh* to:  * prompt for and read a 7-letter word ensuring that exactly 7 characters are read * output whether or not the 7-letter word is a palindrome (reads the same backwards and forwards) | #!/bin/bash  wordcount=0  while [ ${#word} != 7 ]  do  echo "Please enter a 7 letter word: "  read word  done  revw=$(echo $word | rev)  if [ $word = $revw ]  then  echo "The $word is a palindrome"  else  echo "The $word is not a palindrome"  fi |
| 1. Take two screen shots:  * word is a palindrome * word is not a palindrome |  |