Contents

1

Linux

1.1 Resources

Books

1. Unix and Linux System Administration Handbook (Ordered)
2. The Practice of System and Network Administration

Communities

1. Superuser → https://superuser.com/
2. Server fault → https://serverfault.com/
3. Digital Ocean → https://www.digitalocean.com/community/tutorials

Sites

1. Ubuntu → https://help.ubuntu.com/
2. Tutorial Linux → https://tutorialinux.com/

Links

1. https://www.slideshare.net/kavyasri790693/linux-admin-interview-questions
2. http://simplylinuxfaq.blogspot.in/p/linux-system-admin-interview-questions.html
3. https://github.com/kylejohnson/linux-sysadmin-interview-questions/blob/master/test.md
4. https://github.com/chassing/linux-sysadmin-interview-questions#hard

1.2 Users, Passwords & Permissions

Users

1	Adding a user	useradd (single) → newusers (batch mode useradd)
2	Lock an Account	usermod -l <user>
3	New password	passwd <username>
4	Default file permissions	Set UMASK in /etc/login.defs (debians). Takes away the permissions
5	Change Owner & Group	chown
6	Hashed passwords storage	/etc/shadow
7	Change Permissions	chmod Bit mask OGA rwx
8	Delete User	userdel, removing recusively home folder and files → userdel -r

Groups

1	Wheel	Group allowing access to the sudo/su command to become another user or the superuser, for sudo this is enabled with visudo.
2	Add user to a group	usermod -a -G <group> <user> (-a only used with -G, without -a, -G makes the given groups the only additional groups he is a member of)
3	Change users primary group	usermod -g <group> <user>
4	New Group	groupadd <group>
5	All groups on system	getent group
6	chgrp	change the group ownership of a file

Sudo

1. Add a user as a sudoer by using visudo. You can specify users or groups.
2. Common to have a sudo or wheel group and to give that group permissions in visudo
3. Syntax → <user> computerAddress=(<Runas_Alias>) <Command_Alias>
4. You can use a Runas_Alias to define a semi-super user that owns a group of files or processes. Then the user can use sudo to run as that user. Same you can limit the commands that a user can run as sudo with the Command_Alias

5. to give sudo root access use: <user> ALL=(ALL) ALL → root privilages to <user> with use of sudo

1.3 Processes

Process Info

1	PID	Process ID → PID 1 is init, spawns all other ids
2	proc	In /proc → State of running processes in a virtual file system
3	Process types	user → started w/out special permissions, daemon → exist in background, kernel → execute only in 'kernel space'
4	Forked	process being started by a parent process
5	Nice	Priority level [-20 (most) - 19 (least)] → 0 is default. Call with: nice <val> <process>, reset the priority level with: renice <new val> <PID>
6	Process Monitering	Top, ps aux, htop → good tool

Process states

RHEL Doc: https://access.redhat.com/sites/default/files/attachments/processstates_20120831.pdf

1. Runnable
   a) Born or forked
   b) Ready to run or runnable
2. Running
   a) Running in user space or running in kernel space
3. Sleeping
   a) Blocked, Waiting, Sleeping, in an Interruptable sleep, or in an Uninterruptable sleep
   b) The process is sleeping, but it is present in main memory
   c) The process is sleeping, but it is present in secondary memory storage (swap space on disk)
4. Zombie
   a) Terminated or stopped (gone immediately)
   b) **zombie** state if terminated and parent process has not released it yet

Process Signals

1	Send commands	kill PID → kills the process
2	pgrep	Use user or type to find the PID of processes
3	pkill	same as pgrep but it stops the matching PID
4	kill	Send a signal to a process with: kill -s <val> (default is 15) → see man(7) signals for the signals

1.4 Bash Scripting

Shell Variables

1	Set a shell variable from a program output	$(arg)
2	getconf	List system config variables

Pipes & Redirection

1	Pipes	Sends the output of one file into the input of another → cat <filename> — grep <string>
2	Redirect	Use > to overwrite a file, >> to append. Use 1>> for STDOUT & 2>> for STDERR

General Tools

1	curl	Tool for talking over several different protocols
2	wget	Downloads files from an address, same as curl but GNU

1.5 Maintenance

Running Jobs

1	Schedule Jobs (user)	crontab, edit using crontab -e, kept in /var/spool/cron/crontabs, also package specific cron jobs are in /etc/cron.d
2	Schedule Jobs (system)	/etc/crontab
3	at	Run a process at a specified time, accepts HH:MM
4	batch	Run a process when the load drops to a specified level

Backups

Backup Tools: http://www.admin-magazine.com/Articles/Using-rsync-for-Backups
Rsync Snapshots: http://www.mikerubel.org/computers/rsync_snapshots/

- rsync → Remote/Local, Local/Remote, & Local/Local file copying. Sends only the differences between the source & existing files in the destination
  - Use: rsync <options> <source> <destination>
    - Source → Can be files → *.c, or everything in a directory <path name>/, remove the trailing slash to copy the directory.
    - To specify a remote host <computer name> use → <computer name>:<path> as the <source> or <destination>. No : means local only.
  - Options:
    - -a → Archieve mode, saves symbolic links, devices, attributes, permissions, ownership, groups, and is recursive (i.e. -a == -rlptgoD).
    - -t → Transfer files, if file exists, remote-update protocol is used to update the file by sending only the differences
    - -z → Compress before sending
    - –delete → Delete files from the recieving side if not in backup (CAUTION: run –dry-run to see what will be removed first)
    - –progress & -v tells you whats going on
- Backup Types → All can be done using rsync
  - Incrimental → Only record changes from last incrimental backup
  - Differential → Records changes since the last total backup

  - Replica → Just replicate the whole shebang

- Rsync for incrimental backups
  - rsy

1.6 Strings & Searching

Grep

1. Search for a character pattern in a string
2. grep <string> <filename> → returns the lines with the character pattern <string> in file filename
3. Follow directories: grep -r <string> <directory>/*
4. Get the line number → -n
5. Get files with the string → -l
6. Ignore case → -i

Strings

1	cut -d <delim> -f <field1>-<field2>	Break a line on a delim, then take the fields in range, c of chars, b bytes

1.7 Files

Files

1	Types	7 types block special, char spectial, directory, normal file, symbolic link, named pipe, socket
2	diff	Get difference between 2 files or dirs
3	comm	select or reject common lines between files
4	ln	Create a symbolic link
5	link	Create a hard link
6	Find the file's character set	file -i → gives the mime type, search for

File Tools

1	cat	Read a file
2	tac	Read a file backwards
3	Head	Read first few file lines
4	Tail	Read last few file lines
5	read	read from user input → read var → will set the var variable

Find

1. Find a specific file by name find {Starting directory} -name <filename>

2. Finding by type → find <Starting directory> -type <d/f...>

3. Searching depth → find <conditions> -maxdepth <depth>

4. Running a command on all found files → find <conditons> -exec <command> + (the + ends the command)

5. Files by last accessed time → -atime <days_ago or -amin min_ago>

   a) a → accessed, m → modified, c → changed

   b) use -daystart to count from the start of the current day instead of right now

   c) use + for greater than the time, - for less and none for exactly

Finding Stuff

1	Locate (mlocate in suse)	Use updatedb to prepare a database with file locations, then that can be used instead of the slower find
2	which	Shows the full path of (shell) commands (or aliases)
3	whereis	Searches for commands installed and where it is → only for programs no aliases

TAR & ZIP

1	Make a tarball	tar -cpf fileout.tar filename1 filename2..., add p to mantain permissions
2	Extract a tarball	tar -xpf filename.tar (be cautious of 'tarbombs' extract in a directory)
3	tar & gzip	tar -czpf fileout.tar.gz filename1 filename2...
4	Uncompress .tar.gz	tar -xzpf filename.tar.gz
5	Compress to .gz	gzip filename
6	Uncompress .gz	gzip -c filename.gz
7	Compress to .Z	compress filename
8	Uncompress .Z	uncompress filename.Z

1.8 File System

Hierarchy (FHS-V2.3)

Docs: http://www.pathname.com/fhs/pub/fhs-2.3.pdf

1	bin	Essential command binaries
2	boot	Static files of the boot loader → unbootable w/out
3	dev	Device files
4	etc	Host-specific system configuration → must be static, cannot be a binary
5	lib	Essential shared libraries and kernel modules →
6	media	Mount point for removeable media → use lsblk to get the names of these
7	mnt	Mount point for mounting a filesystem temporarily
8	opt	Add-on application software packages
9	sbin	Essential system binaries
10	srv	Data for services provided by this system
11	tmp	Temporary files
12	usr	Secondary hierarchy
13	var	Variable data
14	home (optional)	User home dirs
15	lib<qual> (normally lib64 or lib32, optional)	If multiple library versions are needed like 32 & 64 bit
16	root (optional)	Home dir for root user

Mounting

1	Mounting	mount /dev/<device> destination
2	What disk are mounted	mount
3	Connected disks	lsblk prints out all of the connected devices nicely formatted
4	Mounting on boot	edit /etc/fstab

1.9 Security

SSH hardening

1. Disable SSH protocol 1
2. Reduce the grace time (time to login)
3. Use TCP wrappers (always good to check)
4. Increase key strength (maybe go to 2048-bit keys)
5. Check the defaults and disable a few options

2

Networking

2.1 Resources

Books

1. Beginning Linux Programming (3rd)
2. Unix Network Programming
3. Networking for System Administrators
4. The Practice of System and Network Administration

Sites

1.

Links

1. **Network Questions:** https://github.com/kylejohnson/linux-sysadmin-interview-questions/blob/master/test.md

Sockets

1	Socket	
2	List Sockets	socklist
3	Socket	
4	Socket	

Cider? rfcs?

3

Programming

3.1 GIT

Setup

1	Get a repo	git clone
2	Make a repo	git init
3	Pull an existing repo	Use init or clone the repo then pull
4	Remote repos	git remote → lists the remote repos, git remote add "name" "url"
5	Configuration	git config → complicated, but add email and user with git config –global user.email & user.name

3.2 Terms

Programming

1	Agile	See below 3.2

1. **Agile:** Software development strategy. Values:
   a) **Individuals and Interactions** over processes and tools
      i. Pair programming → 1 station 2 programmers, driver & navigator/observer
      ii. Colocation → Team members in the same area
   b) **Working Software** over comprehensive documentation
   c) **Customer Collaboration** over contract negotiation

d) **Responding to Change** over following a plan