

TP4 — Automatiser l'administration du serveur

BTS SIO 2e année — Linux avancé — Jour 5

Nom : PACHOT Dorian

Classe : BTS SIO 2e année

TP : TP4 — Scripts Bash, cron et systemd

Date : 11 septembre 2026

1. Objectif

Mettre en scripts ce qu'on faisait à la main les jours d'avant, et les faire tourner tout seuls. Sauvegarde, supervision, déploiement, plus une page d'état du serveur.

2. Environnement

Machine	Système	IP	Rôle
poste-hote	Windows	192.168.100.1	Navigateur et SSH
srv-linux	Debian 12	192.168.100.10/24	Tout le TP
cli-linux	Debian 12	DHCP	Client

3. Déroulé

Préparation

```
mkdir -p /home/sio/tp4
sudo mkdir -p /opt/scripts /var/backups/sio /var/log/sio
sudo chown root:root /opt/scripts /var/backups/sio /var/log/sio
```

Instantané VMware « Avant TP4 » pris avant de commencer.

Partie A — les huit scripts

Créés dans /home/sio/tp4, rendus exécutable, passés à bash -n un par un.

```
for f in *.sh; do printf '%-28s ' "$f"; bash -n "$f" && echo "syntaxe OK"; done
01-bienvenue.sh           syntaxe OK
02-tester-fichier.sh      syntaxe OK
03-compter-arguments.sh   syntaxe OK
04-lister-arguments.sh    syntaxe OK
05-table-multiplication.sh syntaxe OK
06-menu.sh                syntaxe OK
07-mot-de-passe.sh        syntaxe OK
08-numeroter-lignes.sh    syntaxe OK
ls -l /home/sio/tp4/
-rwxr-xr-x 1 sio sio 155 11 sept. 09:16 01-bienvenue.sh
-rwxr-xr-x 1 sio sio 564 11 sept. 09:17 02-tester-fichier.sh
-rwxr-xr-x 1 sio sio 203 11 sept. 10:47 03-compter-arguments.sh
-rwxr-xr-x 1 sio sio 187 11 sept. 10:47 04-lister-arguments.sh
-rwxr-xr-x 1 sio sio 335 11 sept. 10:48 05-table-multiplication.sh
-rwxr-xr-x 1 sio sio 500 11 sept. 10:52 06-menu.sh
```

```
-rwxr-xr-x 1 sio sio 382 11 sept. 11:10 07-mot-de-passe.sh
-rwxr-xr-x 1 sio sio 466 11 sept. 11:12 08-numeroter-lignes.sh
```

Les codes de sortie sortent comme prévu, 0 quand ça marche, 1 fichier absent, 2 mauvais usage.

Partie B — menu-admin.sh durci

Copié dans /opt/scripts, réécrit avec en-tête, set -euo pipefail, usage(), log(), trap, mktemp et test root.

```
sudo bash -n /opt/scripts/menu-admin.sh && echo "syntaxe OK"
syntaxe OK
sudo shellcheck /opt/scripts/menu-admin.sh && echo "shellcheck OK"
shellcheck OK
sudo /opt/scripts/menu-admin.sh -h > /dev/null ; echo "code -h : $?"
code -h : 0
sudo /opt/scripts/menu-admin.sh un deux 2> /dev/null ; echo "code 2 args : $?"
code 2 args : 2
sudo tail -5 /var/log/sio/tp4.log
2026-09-11 11:19:19 [INFO ] Choix 3 : services actifs
2026-09-11 11:19:23 [WARN ] Choix invalide : 9
2026-09-11 11:19:24 [INFO ] Sortie demandee par l'utilisateur
```

Ensuite j'ai mis trois défauts exprès dans une copie et shellcheck les a tous vus.

```
rm -f $TMP_TEST SC2086
for f in $(ls /var/log/sio); do echo "$f"; done SC2045
echo "$JOURNAL_" SC2153

apres correction :
shellcheck menu-defaults.sh && echo "shellcheck OK : plus aucun avertissement"
shellcheck OK : plus aucun avertissement
```

Partie C — sauvegarde.sh

```
sudo /opt/scripts/sauvegarde.sh
2026-09-11 11:42:30 [INFO ] ===== Debut de la sauvegarde =====
2026-09-11 11:42:30 [INFO ] Espace disponible : 16019 Mo
2026-09-11 11:42:30 [INFO ] Archivage de : /etc /var/www /srv/samba/commun
2026-09-11 11:42:30 [INFO ] Archive creee : /var/backups/sio/sauvegarde-srv-linux-20260911-1142.tar.gz
2026-09-11 11:42:30 [INFO ] Relecture de l'archive : OK
2026-09-11 11:42:30 [INFO ] Archives supprimees (plus de 7 jours) : 0
2026-09-11 11:42:30 [INFO ] Taille : 700K - Fichiers : 1858 - Duree : 0s - Conservees : 1
2026-09-11 11:42:30 [INFO ] ===== Sauvegarde terminee avec succes =====
Code de sortie : 0
```

Les cinq tests demandés :

```
test 2 : tar -tzf | wc -l -> 1858 entrees, aucun cache sous var/www
test 3 : seuil monte a 999999
[ERROR] Espace insuffisant : 16016 Mo libres, 999999 requis. code 3
test 4 : argument "sept" -> code 2 , argument "3" -> code 0
test 5 : deux lancements -> deux archives, noms differents (la minute change)
verrou : sudo ... & sudo ... -> "Sauvegarde deja en cours." code 3
purge : archive datee de 10 jours -> Archives supprimees : 1
```

Partie D — monitoring.sh

```
sudo /opt/scripts/monitoring.sh ; echo "Code de sortie : $?"
=== RAPPORT DE SUPERVISION - srv-linux - 11/09/2026 13:50 ===
```

POINT DE MONTAGE	TAILLE	UTILISE	USAGE	ETAT
/	19G	2,0G	12%	OK

```

CHARGE                1 MIN      5 MIN      15 MIN      COEURS
moyenne              0.00        0.00        0.00        2

MEMOIRE               TOTALE      DISPO      UTILISEE     ETAT
RAM (Mo)            1931        1483        23%         OK

SERVICE             ETAT
ssh                  actif
apache2              actif
bind9                 actif
smbd                  actif
nfs-kernel-server    actif
isc-dhcp-server       actif

PORT                 ETAT
port 22              en ecoute
port 53              en ecoute
port 80              en ecoute
port 443              en ecoute
port 445              en ecoute
port 2049             en ecoute

=== BILAN : aucune alerte ===
Code de sortie : 0

```

bind9 arrêté, ça donne deux alertes.

```

bind9                  INACTIF
port 53                FERME
=== BILAN : 2 alerte(s) ===
Code de sortie : 2
sudo tail -5 /var/log/sio/monitoring.log
2026-09-11 13:52:45 [ALERTE] Service bind9 inactif
2026-09-11 13:52:45 [ALERTE] Port 53 non en ecoute

```

Test disque avec falloca de 2 Go, on passe de 12 % à 23 %, seuil abaissé à 20, la cellule passe en ALERTE. Fichier supprimé juste après.

Partie E — `deploiement.sh`

```

sudo /opt/scripts/deploiement.sh /srv/sources/site-v2
[INFO ] Version actuelle sauvegardee dans /var/backups/sio/www-20260911-140326
[INFO ] Synchronisation terminee
[INFO ] apache2ctl configtest : OK
[INFO ] Deploiement termine : 2 fichiers publies
Code de sortie : 0
curl -s -o /dev/null -w 'code HTTP : %{http_code}\n' http://www.bts-sio.lan/
code HTTP : 200

```

Puis avec une source qui contient un `.htaccess` invalide :

```

sudo /opt/scripts/deploiement.sh /srv/sources/site-casse
[ERROR] configtest en echec : retour arriere en cours
[INFO ] Contenu precedent restaure depuis /var/backups/sio/www-20260911-140633
[WARN ] Version precedente restauree, Apache recharge
Code de sortie : 4
curl -s http://www.bts-sio.lan/ | grep -i version
<body><h1>Site www.bts-sio.lan - version 2</h1>

```

Le site est resté en version 2, la version cassée n'a jamais été servie.

Partie F — `cron` puis `systemd`

```

sudo crontab -l
# Table de root - TP4 BTS SIO - 11/09/2026
# desactive le 11/09/2026 : remplace par sauvegarde.timer
# 30 2 * * * /opt/scripts/sauvegarde.sh >> /var/log/sio/cron.log 2>&1
*/15 * * * * /opt/scripts/monitoring.sh >> /var/log/sio/cron.log 2>&1
0 23 * * 0 /opt/scripts/rapport-hebdo.sh >> /var/log/sio/cron.log 2>&1
*/5 * * * * /opt/scripts/etat-serveur.sh >> /var/log/sio/cron.log 2>&1

```

Les trois pièges provoqués puis corrigés :

```

piege a : /bin/sh: 1: monitoring.sh: not found
piege b : /bin/sh: 1: /opt/scripts/monitoring.sh: Permission denied
piege c : env sous cron -> 6 variables
          HOME=/root LOGNAME=root PATH=/usr/bin:/bin
          LANG=fr_FR.UTF-8 SHELL=/bin/sh PWD=/root
          env en session -> 21 variables
          PATH=/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games

systemctl list-timers --all --no-pager | grep -i sauvegarde
Sat 2026-09-12 02:30:00 CEST 12h left - - sauvegarde.timer sauvegarde.service
systemctl status sauvegarde.service --no-pager
Loaded: loaded (/etc/systemd/system/sauvegarde.service; static)
Process: 5806 ExecStart=/opt/scripts/sauvegarde.sh (code=exited, status=0/SUCCESS)

```

Partie G — page d'état

```

sudo /opt/scripts/etat-serveur.sh ; echo "Code de sortie : $?"
Code de sortie : 0
sudo ls -l /var/www/www.bts-sio.lan/public_html/etat.html
-rw-r--r-- 1 root www-data 2283 11 sept. 14:28 etat.html
curl -s -o /dev/null -w 'code HTTP : %{http_code}\n' http://www.bts-sio.lan/etat.html
code HTTP : 200
sudo cat /var/log/sio/etat-serveur.log
2026-09-11 14:28:01 [INFO ] Page genereee en 0s, 0 alerte(s) constatee(s)

```

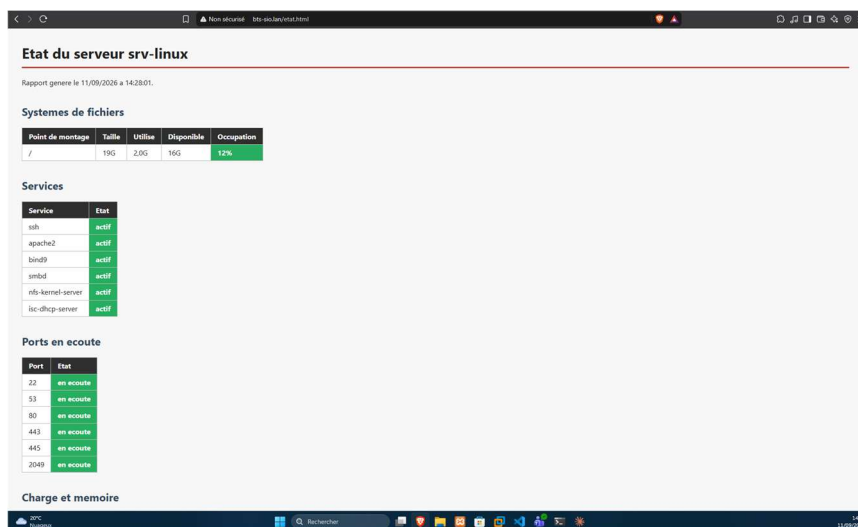


Figure 1 — Page d'état vue depuis le navigateur du poste Windows

Vérification avant rendu

```

services : ssh apache2 bind9 smbd nfs-kernel-server isc-dhcp-server -> tous active
scripts : les 6 de /opt/scripts passent bash -n et shellcheck -> OK
droits : -rwxr-x--- root root sur /opt/scripts/
crontab : aucune ligne de test
disque : /dev/sda1 19G 2,0G 16G 12% / (aucun fichier fallocaete)

```

4. Réponses aux questions

Partie A

Q1. Pourquoi `$USER` affiche `sio` et non `root` sous `sudo`, alors que `whoami` affiche `root` ?

`$USER` c'est une variable d'environnement. `sudo` la réécrit, chez moi elle vaut `root`. Sur une autre config elle pourrait rester à `sio` alors que le script tourne en `root`. `whoami` lui demande au noyau, il peut pas se tromper. Donc on utilise plutôt `$EUID`.

Q2. Les cinq codes de sortie. Différence entre `-e` et `-f`.

0, 0, 0, 1, 2. `-e` c'est vrai dès que le chemin existe, peu importe quoi. `-f` c'est seulement un vrai fichier. `/dev/null` existe mais c'est pas un fichier normal. On teste `-e` avant sinon on dit que c'est pas un fichier alors qu'en fait ça existe pas du tout.

Q3. Différence entre `"$@"` et `"$*"`.

`"$@"` garde chaque argument séparé. `"$*"` colle tout en un seul mot. Pour relayer à une autre commande on prend `"$@"`, sinon `tar` recevrait un seul nom de fichier qui existe pas.

Q4. Pourquoi `for arg in "$@"` et pas `$@` ?

Avec les guillemets, mon fichier.txt fait un tour de boucle. Sans, le shell coupe sur l'espace et ça fait deux tours. Je l'ai vérifié, ça affiche [mon] puis [fichier.txt].

Q5. Pourquoi `printf` plutôt que `echo` ? Et sans le `\n` ?

`printf` aligne les colonnes tout seul avec `%3d`, `echo` il faut mettre les espaces à la main. Sans le `\n` tout se met sur une seule ligne.

Q6. Pourquoi `4|q/Q` ? À quoi sert le `;;` ?

`case` compare des motifs, la barre verticale fait partie de la syntaxe. Si j'écris « 4 ou q ou Q » il prend ça pour un seul motif. Le `;;` ferme la branche, sans lui `bash` continue et plante sur la parenthèse d'après.

ligne 12: erreur de syntaxe pres du symbole inattendu <<) >>

Q7. `read -rp` au lieu de `read -rsp` ? Le mot de passe en clair ?

Avec `-rp` ce qu'on tape s'affiche à l'écran. Le `-s` le cache. Le mot de passe en clair dans le script c'est nul parce que tout le monde qui lit le fichier le connaît, et il part dans les sauvegardes. Mieux : laisser `sudo` ou `PAM` vérifier, ou comparer un hash.

Q8. `IFS=` et `-r` ? Pourquoi pas `cat | while` ?

`IFS=` garde les espaces du début de ligne, `-r` garde les antislashes. Quand j'enlève `IFS=`, la ligne indentée perd ses quatre espaces. `cat | while` c'est déconseillé parce que le tube met la boucle dans un sous-shell et mes compteurs seraient perdus.

Partie B

Q9. La fonction `log()` : `shift`, `tee -a`, le format de date.

Le `shift` enlève le niveau des arguments pour que `"$*"` soit que le message. `tee -a` écrit à l'écran et dans le journal en même temps, une simple redirection écrirait que dans le fichier. Le format `%Y-%m-%d` met l'année d'abord donc trier par ordre alphabétique revient à trier par date.

Q10. Le script sans `sudo`.

-bash: /opt/scripts/menu-admin.sh: Permission non accordée
code : 126

Chez moi ça donne pas 3 mais 126, parce que les droits 750 root:root font que sio peut même pas lancer le fichier. Le test se met avant tout le reste parce que la suite a besoin de root. Le message part sur >&2 pour pas se mélanger avec la sortie normale.

Q11. Les trois options de set.

-e arrête dès qu'une commande rate. -u une variable pas définie devient une erreur. pipefail fait échouer le tube si n'importe quelle commande dedans rate. Celui qui pose problème c'est -e avec un grep qui trouve rien, parce que grep renvoie 1 et c'est normal. La parade c'est `|| true` derrière, ou mettre la commande dans un if.

Q12. Les trois avertissements shellcheck.

```
SC2086 : Double quote to prevent globbing and word splitting.
  risque : rm -f $TMP_TEST sur "mon fichier.txt" supprime deux fichiers
  correction : rm -f "$TMP_TEST"

SC2045 : Iterating over ls output is fragile. Use globs.
  risque : un nom avec une espace fait deux tours de boucle
  correction : for f in /var/log/sio/*

SC2153 : Possible misspelling: JOURNAL_ may not be assigned.
  risque : la variable vaut rien, le script écrit dans un chemin vide
  correction : echo "$JOURNAL"
```

Sur le troisième j'ai eu SC2153 et pas SC2154 comme dans l'énoncé, parce que le nom ressemble à une variable qui existe déjà.

Q13. Pourquoi bash -n a rien vu ?

Parce que les trois défauts c'est du bash correct, c'est de la logique pas de la syntaxe. bash -n regarde que la syntaxe, bash -x montre ce qui s'exécute vraiment, shellcheck lit le sens du code sans l'exécuter. Un nom de variable mal écrit il n'y a que shellcheck qui peut le voir.

Partie C

Q14. Le journal d'une exécution. Combien d'étapes ? Les niveaux.

Sept étapes : début, espace, sources, archive créée, relecture, purge, résumé. INFO c'est le déroulement normal, WARN c'est pas grave mais c'est bizarre (tar qui râle parce qu'un fichier a bougé), ERROR c'est que la sauvegarde a pas eu lieu.

Q15. La commande pour l'espace libre. /var est-il une partition à part ?

```
df -BM --output=avail /var/backups/sio | tail -1
16018M
```

-BM pour avoir des mégaoctets, --output=avail pour garder que la colonne du dispo, tail -1 pour virer l'en-tête. On mesure là où on écrit parce que c'est ce disque-là qui va se remplir. Chez moi /var est pas une partition à part, df donne le même /dev/sda1 que / et findmnt /var trouve rien.

Q16. Pourquoi le code 3 et pas 1 ?

3 ça veut dire qu'il a même pas essayé. 1 ça voudrait dire qu'il a essayé et que ça a raté. C'est pas la même chose à faire derrière : sur un 3 on libère de la place, sur un 1 on va chercher un problème de tar.

Q17. Le nombre de fichiers et le grep cache.

1858 entrées. Le grep cache trouve 31 lignes mais elles sont toutes dans etc/, des modules Apache et ld.so.cache. Rien sous var/www. L'exclusion c'est --exclude='/var/www/*/cache', avec le chemin.

Si j'avais mis --exclude=cache tout court, ça aurait aussi viré /etc/cache de la sauvegarde sans que je le voie.

Q18. *Les deux archives ont-elles des noms différents ? Le format de date.*

Oui, la minute a changé. Le format %Y%m%d met l'année d'abord donc les classe dans le bon ordre. Par contre si je lance deux fois dans la même minute j'ai le même nom et la deuxième écrase la première, je l'ai vu dans le journal à 11:51.

Q19. *Le verrou flock.*

```
Sauvegarde déjà en cours.  
Code : 3
```

flock pose un verrou sur un descripteur de fichier, et c'est le noyau qui le tient. Du coup il se libère tout seul quand le processus se termine, même si ça plante. Un fichier PID écrit à la main il reste après un plantage et le script refuse de démarrer pour toujours.

Partie D

Q20. *Le rapport. Combien de lignes ? Le format %-28s.*

15 lignes de détail. Le 28 c'est la largeur de la colonne, le signe - aligne à gauche. Sans le -, ça aligne à droite et les colonnes d'état seraient décalées à chaque ligne vu que les noms de service font pas la même longueur.

Q21. *bind9 avant et après. Pourquoi is-active --quiet dans un if ?*

Avant : bind9 actif, code 0. Après : bind9 INACTIF, code 2 (le service plus le port 53). On met is-active dans le if parce que if teste un code de retour, et comparer la sortie à « active » c'est fragile, systemctl peut répondre activating ou failed et le texte change selon la version.

Q22. *Le test d'un port, et les options de ss.*

```
ECOUTE=$(ss -tulpn)  
grep -qE "[[:.]]${port}[[[:space:]]]" <<< "$ECOUTE"
```

-t tcp, -u udp, -l que ce qui écoute, -n les numéros sans traduire, -p le processus. Je stocke dans une variable pour lancer ss une seule fois au lieu de six, sinon ce sont six photos prises à six moments différents.

Q23. *Le signe % et la comparaison.*

```
[ : 73% : nombre entier attendu comme expression
```

L'expansion c'est \${valeur%}. Le premier % coupe à la fin de la chaîne, le \% c'est le caractère qu'on veut enlever. La comparaison rate parce que -ge veut un entier et 73% c'est une chaîne.

Q24. *Le code de sortie et le plafond à 255.*

Avec le seuil à 10 j'ai une alerte et le code vaut 1, donc oui ça correspond. On plafonne parce qu'un code de sortie tient sur un octet.

```
exit 255 donne : 255  
exit 256 donne : 0  
exit 300 donne : 44
```

Le pire c'est exit 256 qui donne 0 : le superviseur croirait que tout va bien alors qu'il y a 256 problèmes.

Q25. *Pourquoi pas de -e ici ?*

Parce qu'un script de supervision doit contrôler tous les points même si un est en panne. Chez moi bc était pas installé et la ligne de la charge a planté, mais comme il n'y a pas de -e j'ai quand même eu les services et les ports. Avec -e j'aurais rien su. -u et pipefail on les garde parce que eux ils attrapent des bugs du script, pas des états du serveur.

Partie E

Q26. Le journal du déploiement fautif. Quand décide-t-il de revenir en arrière ?

```
[INFO ] Version actuelle sauvegardee dans /var/backups/sio/www-20260911-140633
[INFO ] Synchronisation terminee
[INFO ] Proprietaire root:www-data et permissions 750 appliques
AH00526: Syntax error on line 1 of .../.htaccess:
Invalid command 'DirectiveQuiNexistePas'
[ERROR] configtest en echec : retour arriere en cours
[INFO ] Contenu precedent restaure
[WARN ] Version precedente restauree, Apache recharge
Code de sortie : 4
```

Il décide juste après configtest, donc après avoir copié les fichiers mais avant le reload. C'est le bon moment : Apache tourne encore avec l'ancienne config en mémoire. Le retour arrière fait cinq commandes : rsync depuis la copie, chown, chmod, un nouveau configtest, puis reload.

Q27. La barre oblique finale et --delete.

```
avec la barre : images index.html
sans la barre : site-v2/images site-v2/index.html
```

Avec la barre on copie le contenu du dossier, sans la barre on copie le dossier lui-même et ça crée un sous-dossier dans la cible. --delete supprime dans la cible ce qui est plus dans la source, sinon les vieilles pages resteraient en ligne pour toujours. C'est aussi pour ça qu'il faut la sauvegarde avant, une faute de frappe sur la source viderait le site.

Q28. Les droits root:www-data et 750.

root a tout, www-data peut lire et traverser, les autres rien du tout (depuis sio je peux même plus faire un ls). Apache tourne en www-data et il a juste besoin de lire pour servir une page, l'écriture c'est le boulot du déploiement. Le -R 750 met le bit x sur les fichiers html alors qu'ils ont pas à être exécutables. Ce serait mieux de faire un find -type d en 750 et un find -type f en 640.

Q29. Pourquoi un code 4 spécifique ?

Nominal 0, fautif 4. Le 4 dit que ça a raté mais que le site tourne toujours sur l'ancienne version. Un simple 1 dirait juste qu'il y a eu une erreur sans dire dans quel état est le serveur. Pour de la CI ou du cron ça permet de pas déclencher une alerte d'incident alors que le site marche.

Partie F

Q30. La crontab, les champs, et l'ordre de la redirection.

30 2 * * * : minute 30, heure 2, tous les jours. */15 * * * * : toutes les 15 minutes, à toutes les heures. On écrit >> fichier 2>&1 parce que 2>&1 envoie la sortie d'erreur là où pointe la sortie standard à ce moment-là, donc il faut d'abord rediriger vers le fichier.

```
ordre correct : le fichier contient les deux lignes
ordre inverse : l'erreur part a l'ecran, le fichier a que la sortie standard
```

Q31. Piège a, le chemin relatif.

```
/bin/sh: 1: monitoring.sh: not found
```

Trouvé dans `/var/log/sio/piege-a.log` grâce au `2>&1`. Chez moi ça marche pas non plus quand je le tape dans ma session, `/opt/scripts` est dans aucun des deux `PATH`. Ce qui marcherait en session et pas sous cron c'est ce qui est dans `/usr/local/bin`. Les deux corrections : chemin absolu dans la crontab, ou déclarer `PATH` en haut de la crontab.

Q32. Piège b, le bit x.

```
/bin/sh: 1: /opt/scripts/monitoring.sh: Permission denied
```

```
126 = fichier trouve mais pas executable
127 = commande introuvable
```

Au piège a il trouve pas le fichier, au piège b il le trouve mais il a pas le droit de l'exécuter. Un problème de chemin contre un problème de droits.

Q33. Piège c, l'environnement.

```
HOME=/root
LOGNAME=root
PATH=/usr/bin:/bin
LANG=fr_FR.UTF-8
SHELL=/bin/sh
PWD=/root
```

```
cron : 6 variables      session : 21 variables
```

Trois qui manquent sous cron : `TERM`, donc un script qui utilise `tput` planterait. `USER`, donc un script qui écrit « genere par \$USER » écrirait rien, et avec `set -u` il s'arrêterait. `XDGRUNTIME_DIR`, donc un `systemctl --user` trouverait pas sa socket. Les deux façons de s'en sortir : tout déclarer dans le script avec des chemins absolus, ou déclarer `PATH` et compagnie en tête de crontab.

Q34. Les deux unités, les colonnes du timer, et activer le .service.

```
/etc/systemd/system/sauvegarde.service
[Unit]
Description=Sauvegarde quotidienne du serveur SIO
Wants=network-online.target
After=network-online.target

[Service]
Type=oneshot
ExecStart=/opt/scripts/sauvegarde.sh
User=root
Nice=10
IOSchedulingClass=idle

/etc/systemd/system/sauvegarde.timer
[Unit]
Description=Declenche la sauvegarde SIO tous les jours a 2 h 30

[Timer]
OnCalendar=*-*-* 02:30:00
Persistent=true
Unit=sauvegarde.service

[Install]
WantedBy=timers.target
```

`NEXT` vaut `Sat 2026-09-12 02:30:00`, `LEFT` 12h left, `LAST` est vide parce que le timer a jamais encore déclenché (mon lancement de 14h23 je l'ai fait à la main). Quand j'essaie d'activer le `.service`, `systemd` refuse : il a pas de section `[Install]`, il est marqué `static`. Donc ça programme rien du tout.

Q35. Tableau comparatif cron / timer systemd.

Critère	cron	Timer systemd	Comment j'ai vu
Mise en place	1 ligne	2 fichiers + daemon-reload + enable	crontab -l / list-timers
Lisibilité	5 champs à décoder	OnCalendar lisible	systemd-analyze calendar
Rattrapage si éteint	non	oui, Persistent=true	le fichier .timer
Sortie du script	là où je l'ai redirigée	direct dans le journal	journalctl -u sauvegarde.service
Code de sortie gardé	non	oui, status=0/SUCCESS	systemctl status
Dépendance réseau	rien	After=network-online.target	systemctl cat
Priorité CPU/disque	rien	Nice et IOSchedulingClass	systemctl cat
Double lancement	à faire avec flock	natif	systemctl status

Je garderais cron pour un truc simple sur une machine allumée tout le temps, une ligne suffit et tout le monde sait la lire. Je passerais au timer dès qu'il faut du rattrapage, une dépendance au réseau ou garder le code de sortie, donc pour la sauvegarde de nuit.

Partie G

Q36. La page d'état : les fonctions, la couleur, l'écriture atomique, l'horodatage.

(a) Cinq fonctions d'affichage : entete_html écrit le début du document avec le style, tableau_disques lit df, tableau_services teste les six services, tableau_ports cherche les six ports, pied_html écrit la charge, la mémoire et ferme le document. Le corps principal c'est juste les cinq appels à la suite.

(b) La couleur vient de couleur_pourcent, qui compare à deux seuils.

```
couleur_pourcent() {
    local v="$1"
    if [ "$v" -ge "$SEUIL_ROUGE" ]; then echo "#e74c3c"
    elif [ "$v" -ge "$SEUIL_ORANGE" ]; then echo "#e67e22"
    else echo "#27ae60"
    fi
}
```

Chez moi on est à 12 % donc c'est vert.

(c) L'écriture atomique c'est ces deux lignes :

```
124:TMP=$(mktemp "$(dirname "$CIBLE")/etat.XXXXXX")
137:mv "$TMP" "$CIBLE"
```

Le script écrit tout dans un fichier temporaire puis le renomme d'un coup. Comme le temporaire est dans le même dossier, mv fait juste un renommage et les lecteurs voient soit l'ancien fichier soit le nouveau. Sans ça, quelqu'un qui recharge pendant la génération recevrait une page coupée au milieu.

(d) Oui, ça change bien.

```
a 14:31:06 -> Rapport genere le 11/09/2026 a 14:28:01.
a 14:37:07 -> Rapport genere le 11/09/2026 a 14:35:01.
```

5. Fichiers

Les scripts sont livrés dans l'archive, aux chemins suivants.

```
/home/sio/tp4/01-bienvenue.sh ... 08-numeroter-lignes.sh (755 sio:sio)
/opt/scripts/menu-admin.sh (750 root:root)
```

```

/opt/scripts/sauvegarde.sh (750 root:root)
/opt/scripts/monitoring.sh (750 root:root)
/opt/scripts/deploiement.sh (750 root:root)
/opt/scripts/etat-serveur.sh (750 root:root)
/opt/scripts/rapport-hebdo.sh (750 root:root)
/etc/systemd/system/sauvegarde.service
/etc/systemd/system/sauvegarde.timer

```

/opt/scripts/sauvegarde.sh

```

#!/bin/bash
#=====
# Nom      : sauvegarde.sh
# Objet    : Archive /etc, /var/www et /srv/samba/commun dans
#           /var/backups/sio, puis purge les archives de plus
#           de N jours.
# Auteur   : sio <sio@bts-sio.lan>
# Cree le  : 11/09/2026      Version : 1.0
# Usage    : sudo /opt/scripts/sauvegarde.sh [jours_retention]
# Sorties  : 0 succes, 1 archive en echec ou illisible,
#           2 mauvais usage (argument non numerique),
#           3 espace insuffisant ou droits insuffisants
#=====
set -euo pipefail
#
# ----- Parametres (modifiables ici uniquement) -----
SOURCES=(/etc /var/www /srv/samba/commun)
DESTINATION="/var/backups/sio"
JOURNAL="/var/log/sio/sauvegarde.log"
RETENTION="${1:-7}"
ESPACE_MINI_MO=1024
EXCLUSION="/var/www/*/cache"
DEBUT=$SECONDS
#
# ----- Fonctions -----
log() {
    local niveau="$1"; shift
    printf '%s [%-5s] %s\n' "$(date '+%Y-%m-%d %H:%M:%S')" "$niveau" "$*" \
        | tee -a "$JOURNAL"
}
usage() {
    cat <<EOF
Usage : $0 [jours_retention]
    jours_retention  nombre de jours de conservation (default : 7)

Codes de sortie : 0 succes, 1 archive en echec, 2 mauvais usage,
3 espace insuffisant ou droits insuffisants.
EOF
}
nettoyer() {
    # supprime une archive vide laissee par une interruption
    # shellcheck disable=SC2317 # appelee indirectement par trap
    if [ -f "$ARCHIVE" ] && [ ! -s "$ARCHIVE" ]; then rm -f "$ARCHIVE"; fi
}
#
# ----- Prerequis -----
if [ "$EUID" -ne 0 ]; then
    echo "Ce script doit etre lance en root (sudo)." >&2
    exit 3
fi
if [ "$#" -gt 1 ]; then
    usage >&2
    exit 2
fi
if ! [[ "$RETENTION" =~ ^[0-9]+$ ]]; then

```

```

    echo "Retention non numerique : $RETENTION" >&2
    usage >&2
    exit 2
fi
mkdir -p "$DESTINATION" "${dirname "$JOURNAL")"
ARCHIVE="$DESTINATION"/sauvegarde-$(hostname)-$(date +%Y%m%d-%H%M).tar.gz"
trap nettoyer EXIT INT TERM
exec 9> /var/lock/sauvegarde.lock
flock -n 9 || { echo "Sauvegarde deja en cours." >&2; exit 3; }
log INFO "==== Debut de la sauvegarde ====="
#
# ----- Controle d'espace -----
LIBRE_MO=$(df -BM --output=avail "$DESTINATION" | tail -1 | tr -dc '0-9')
if [ "$LIBRE_MO" -lt "$ESPACE_MINI_MO" ]; then
    log ERROR "Espace insuffisant : ${LIBRE_MO} Mo libres, ${ESPACE_MINI_MO} requis."
    exit 3
fi
log INFO "Espace disponible : ${LIBRE_MO} Mo"
#
# ----- Archivage -----
log INFO "Archivage de : ${SOURCES[*]}"
if tar -czf "$ARCHIVE" --exclude="$EXCLUSION" --warning=no-file-changed "${SOURCES[@]}" 2>>
"$JOURNAL"; then
    log INFO "Archive creee : $ARCHIVE"
else
    CODE=$?
    # tar renvoie 1 pour un simple avertissement, 2 pour une vraie erreur
    if [ "$CODE" -gt 1 ]; then log ERROR "Echec de tar (code $CODE)."; exit 1; fi
    log WARN "tar a signale des avertissements (code $CODE), archive conservee."
fi
#
# ----- Controle d'integrite -----
if ! tar -tzf "$ARCHIVE" > /dev/null 2>&1; then
    log ERROR "Archive illisible, suppression."
    rm -f "$ARCHIVE"
    exit 1
fi
log INFO "Relecture de l'archive : OK"
#
# ----- Purge et resume -----
NB_PURGEES=$(find "$DESTINATION" -maxdepth 1 -name 'sauvegarde-*.tar.gz' \
    -mtime +"$RETENTION" -print -delete | wc -l)
log INFO "Archives supprimees (plus de ${RETENTION} jours) : ${NB_PURGEES}"
TAILLE=$(du -h "$ARCHIVE" | cut -f1)
NB_FICHIERS=$(tar -tzf "$ARCHIVE" | wc -l)
NB_RESTANTES=$(find "$DESTINATION" -maxdepth 1 -name 'sauvegarde-*.tar.gz' | wc -l)
log INFO "Archive : $ARCHIVE"
log INFO "Taille : ${TAILLE} - Fichiers : ${NB_FICHIERS} - Duree : $((SECONDS - DEBUT))s -
Conservees : ${NB_RESTANTES}"
log INFO "==== Sauvegarde terminee avec succes ====="
exit 0

```

/opt/scripts/monitoring.sh

```

#!/bin/bash
#=====
# Nom      : monitoring.sh
# Objet    : Controle des systemes de fichiers, des services,
#            des ports en ecoute, de la charge et de la memoire.
# Auteur   : sio <sio@bts-sio.lan>
# Cree le  : 11/09/2026          Version : 1.0
# Usage    : sudo /opt/scripts/monitoring.sh [seuil_disque]
# Sorties  : nombre d'alertes detectees (0 = tout va bien),
#            plafonne a 255 ; 2 mauvais usage
#=====
set -uo pipefail          # pas de -e : on poursuit malgre une alerte

```

```

#
# ----- Parametres (modifiables ici uniquement) -----
SEUIL_DISQUE="${1:-80}"
JOURNAL="/var/log/sio/monitoring.log"
SERVICES=(ssh apache2 bind9 smbd nfs-kernel-server isc-dhcp-server)
PORTS=(22 53 80 443 445 2049)
ALERTE=0
#
# ----- Fonctions -----
usage() {
    cat <<EOF
Usage : $0 [seuil_disque]
    seuil_disque    pourcentage d'occupation declenchant une alerte,
                    entier entre 1 et 99 (default : 80)

Sortie : nombre d'alertes detectees, plafonne a 255. 2 si mauvais usage.
EOF
}
alerte() {
    ALERTE=$(( ALERTE + 1 ))
    printf '%s [ALERTE] %s\n' "$(date '+%Y-%m-%d %H:%M:%S')" "$*" >> "$JOURNAL"
}
#
# ----- Prerequis -----
if [ "$#" -gt 1 ]; then
    usage >&2
    exit 2
fi
if ! [ [ "$SEUIL_DISQUE" =~ ^[0-9]+$ ] || [ "$SEUIL_DISQUE" -lt 1 ] || [ "$SEUIL_DISQUE" -gt 99 ]; then
    echo "Seuil invalide : $SEUIL_DISQUE (entier attendu entre 1 et 99)" >&2
    usage >&2
    exit 2
fi
mkdir -p "$(dirname "$JOURNAL")"
#
# ----- Rapport -----
printf '=== RAPPORT DE SUPERVISION - %s - %s ==\n\n' \
    "$(hostname)" "$(date '+%d/%m/%Y %H:%M')"
#
# ----- 1. Systemes de fichiers -----
printf '%-28s %-10s %-10s %-10s %s\n' "POINT DE MONTAGE" "TAILLE" "UTILISE" "USAGE" "ETAT"
while read -r taille utilise pourcent point; do
    valeur="${pourcent%}"
    if [ "$valeur" -ge "$SEUIL_DISQUE" ]; then
        etat="ALERTE"
        alerte "Systeme de fichiers $point occupe a ${valeur}%"
    else
        etat="OK"
    fi
    printf '%-28s %-10s %-10s %-10s %s\n' "$point" "$taille" "$utilise" "${valeur}%"
done < <(df -h -t ext4 -t xfs --output=target,size,used,pcent | tail -n +2 | awk '{print $2, $3, $4, $1}')
#
# ----- 2. Charge et memoire -----
CHARGE1=$(cut -d ' ' -f1 /proc/loadavg)
CHARGE5=$(cut -d ' ' -f2 /proc/loadavg)
CHARGE15=$(cut -d ' ' -f3 /proc/loadavg)
CPUS=$(nproc)
printf '\n%-28s %-10s %-10s %-10s %s\n' "CHARGE" "1 MIN" "5 MIN" "15 MIN" "COEURS"
printf '%-28s %-10s %-10s %-10s %s\n' "moyenne" "$CHARGE1" "$CHARGE5" "$CHARGE15" "$CPUS"
if [ "$(echo "$CHARGE1 > $CPUS" | bc -l)" -eq 1 ]; then
    alerte "Charge 1 min $CHARGE1 superieure au nombre de coeurs ($CPUS)"
fi

```

```

MEM_TOTAL=$(free -m | awk '/^Mem:/ {print $2}')
MEM_DISPO=$(free -m | awk '/^Mem:/ {print $7}')
MEM_PCT=$(( (MEM_TOTAL - MEM_DISPO) * 100 / MEM_TOTAL ))
printf '\n%-28s %-10s %-10s %-10s %s\n' "MEMOIRE" "TOTALE" "DISPO" "UTILISEE" "ETAT"
if [ "$MEM_PCT" -ge 90 ]; then
    alerte "Memoire utilisee a ${MEM_PCT}%"
    printf '%-28s %-10s %-10s %-10s %s\n' "RAM (Mo)" "$MEM_TOTAL" "$MEM_DISPO"
    "${MEM_PCT}%" "ALERTE"
else
    printf '%-28s %-10s %-10s %-10s %s\n' "RAM (Mo)" "$MEM_TOTAL" "$MEM_DISPO"
    "${MEM_PCT}%" "OK"
fi
#
# ----- 3. Services -----
printf '\n%-28s %s\n' "SERVICE" "ETAT"
for service in "${SERVICES[@]}; do
    if systemctl is-active --quiet "$service"; then
        printf '%-28s %s\n' "$service" "actif"
    else
        printf '%-28s %s\n' "$service" "INACTIF"
        alerte "Service $service inactif"
    fi
done
#
# ----- 4. Ports en ecoute -----
ECOUTE=$(ss -tulpn 2>/dev/null || ss -tulnH)
printf '\n%-28s %s\n' "PORT" "ETAT"
for port in "${PORTS[@]}; do
    if grep -qE "[:.]\${port}[:space:]" <<< "$ECOUTE"; then
        printf '%-28s %s\n' "port $port" "en ecoute"
    else
        printf '%-28s %s\n' "port $port" "FERME"
        alerte "Port $port non en ecoute"
    fi
done
#
# ----- Bilan -----
if [ "$ALERTES" -eq 0 ]; then
    printf '\n=== BILAN : aucune alerte ===\n'
else
    printf '\n=== BILAN : %s alerte(s) ===\n' "$ALERTES"
fi
# un code de sortie ne peut pas dépasser 255
if [ "$ALERTES" -gt 255 ]; then ALERTES=255; fi
exit "$ALERTES"

```

Les six autres scripts sont dans l'archive. Les deux unités systemd et la crontab sont recopiées plus haut, aux questions 34 et 30.

6. Difficultés

bc pas installé. La comparaison de la charge plantait avec « bc : commande introuvable ». Bash calcule que sur des entiers et la charge c'est un décimal. Installé avec apt.

rsync pas installé non plus. Le déploiement s'est arrêté sur « rsync absent. » avec le code 3, donc le contrôle de prérequis a fait son boulot. Installé aussi.

shellcheck m'a sorti SC2317 sur la fonction nettoyer() de sauvegarde.sh, il croit que le code est mort parce qu'elle est appelée par trap. C'est un faux positif, je l'ai désactivé sur la ligne seulement.

L'énoncé annonce le code 3 quand on lance menu-admin.sh sans sudo, mais avec les droits 750 root:root imposés on obtient 126 : sio peut même pas lancer le fichier. J'ai fait une copie lisible dans /tmp pour voir le vrai code 3.

apache2ctl configtest lit pas les .htaccess, donc le test du déploiement fautif marchait pas. J'ai ajouté un IncludeOptional dans le fichier de site le temps du test, après l'avoir sauvegardé, et je l'ai remis après.

Le tube systemctl | head cassait pas le script malgré pipefail, parce que la sortie tient dans le tampon du tube. J'ai refait la démo avec false | true et yes | head qui donnent 1 et 141.

7. Conclusion

Tout ce qui était demandé tourne. La sauvegarde se vérifie elle-même et refuse un double lancement, la supervision sort un code égal au nombre d'alertes, le déploiement est revenu en arrière tout seul et le site est jamais tombé, et la page d'état se régénère toutes les cinq minutes.

Ce qui reste : les codes de sortie de monitoring.sh se chevauchent entre le nombre d'alertes et le mauvais usage, et l'horodatage à la minute permet encore d'écraser une archive. La bibliothèque commune du bonus est pas faite.

Ce que je retiens : un script se juge sur ce qu'il fait quand ça rate. Et le journal de cron prouve juste qu'une tâche a été lancée, jamais qu'elle a réussi, donc c'est au script de journaliser lui-même.