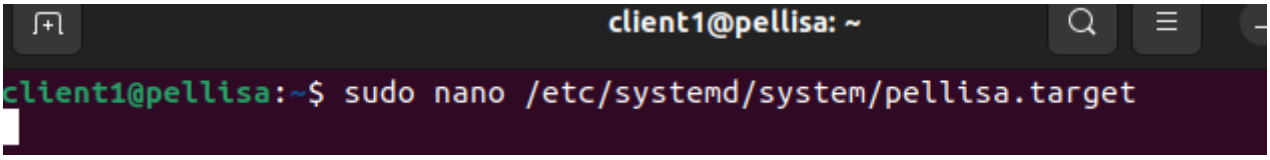
En aquesta pràctica treballarem amb **systemd**.

Crearem un **target** personalitzat per agrupar serveis.

També configurarem un **servei** que executarà un script.

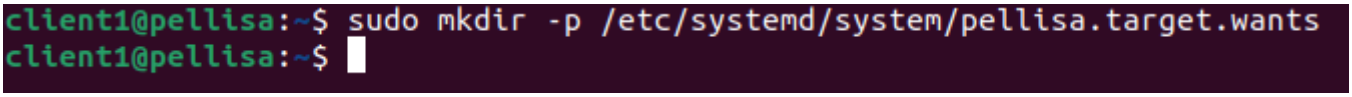
L'script realitzarà una acció simple per comprovar el funcionament.

Finalment, activarem i provarem tot el conjunt dins del sistema.

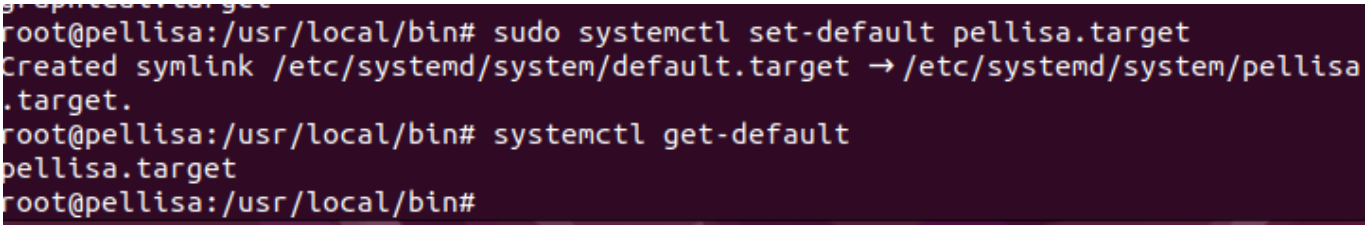


Primer que tot crearem el Target.

He creat un target que s'activa quan tot està llest: xarxa, usuaris i escriptori. Així controlo que els meus serveis s'inicïen en l'ordre correcte i amb totes les dependències satisfetes.



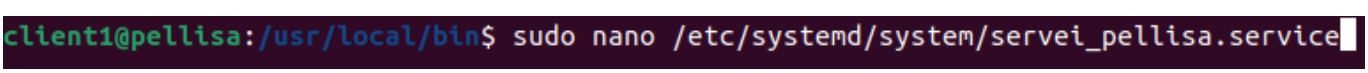
Aqui creem aquest directori perquè és on systemd busca els serveis que han d'activar-se quan el teu target `pellisa.target` s'iniciï. Els enllaços simbòlics dels serveis que vulguis que s'executin amb el teu target aniran aquí dins.



Hem configurat el sistema perquè utilitzi el nostre target personalitzat com a target per defecte.

1. `systemctl set-default pellisa.target` - Crea un enllaç simbòlic per definir el nostre target com a predeterminat
2. `systemctl get-default` - Verifica que efectivament `pellisa.target` és ara el target per defecte

Ara el sistema sempre arrencarà amb la nostra configuració personalitzada.



Ara crearem el servei que tindrem associat al target.

```
  GNU nano 6.2                /etc/systemd/system/servei_pellisa.service *
[Unit]
Description=Servei que executa el meu script curiós jejeje
After=network.target
Wants=network.target

[Service]
Type=simple
ExecStart=/usr/local/bin/script-pellisa.sh
Restart=on-failure

[Install]
WantedBy=pellisa.target
```

Aqui veiem el servei systemd que executara el meu script personalitzat. El servei s'assegura que:

- S'inicia després que la xarxa estigui activa
- Es reinicia automàticament si falla
- Està gestionat pel meu target `pellisa.target`
- És de tipus simple per executar el script directament

```
root@pellisa:/usr/local/bin# systemctl enable servei_pellisa.service
Created symlink /etc/systemd/system/pellisa.target.wants/servei_pellisa.service → /etc/system
d/system/servei_pellisa.service.
root@pellisa:/usr/local/bin#
```

He activat el servei perquè s'iniciï automàticament amb el target `pellisa.target`. Systemd ha creat un enllaç simbòlic dins de la carpeta `pellisa.target.wants`, que és exactament per això que havíem creat aquest directori abans.

```
root@pellisa:/usr/local/bin# systemctl daemon-reload
root@pellisa:/usr/local/bin#
```

He recarregat la configuració de systemd perquè detecti el nou servei creat. Això és necessari cada vegada que es crea o modifica un fitxer de servei perquè systemd l'incorpori sense necessitat de reiniciar el sistema.

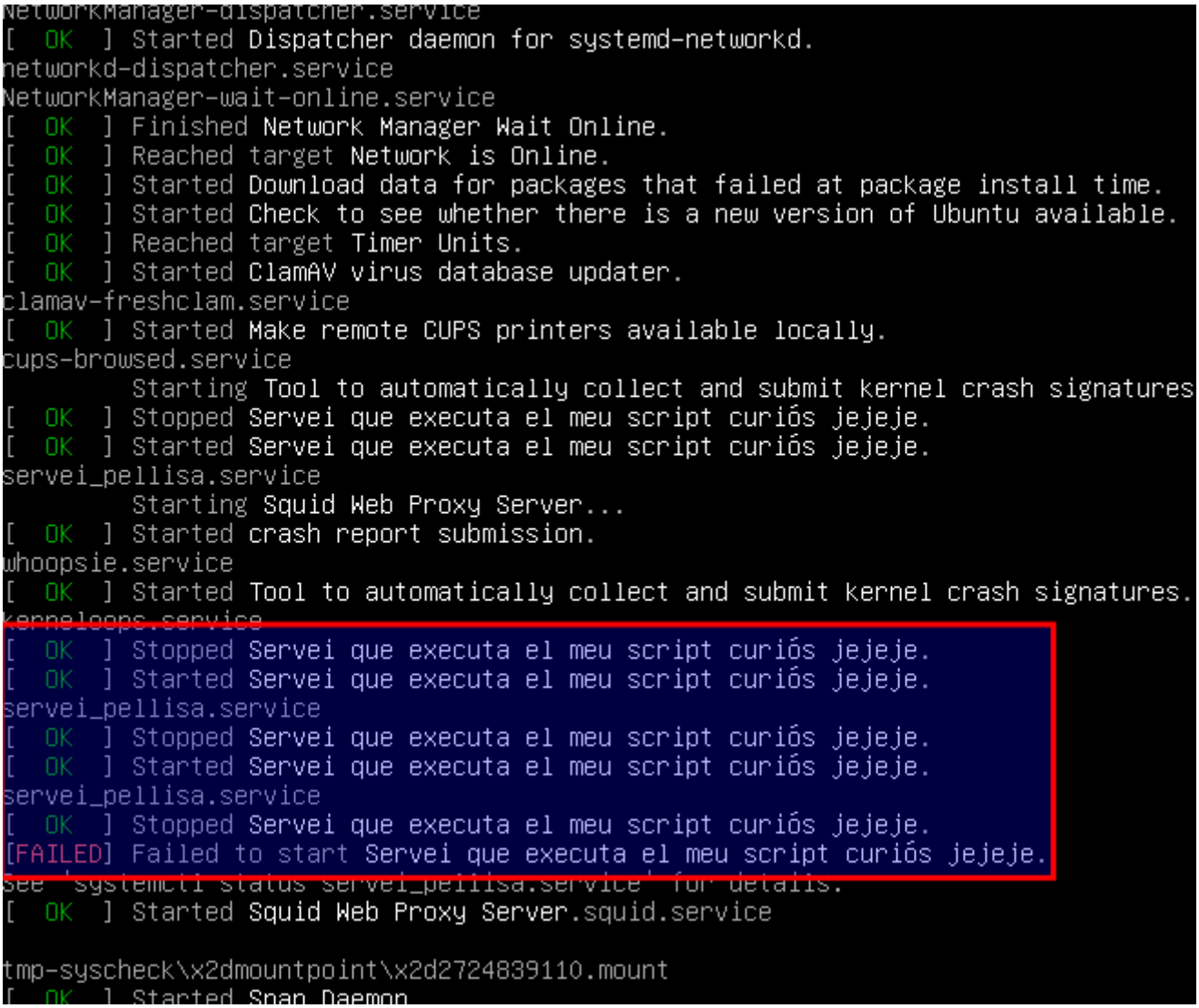
```bash
#!/bin/bash


CARPETA="/home/client1/Escritorio/privat_pellisa"
mkdir -p "$CARPETA"

DATA=$(date '+%Y-%m-%d_%H-%M-%S')
FITXER="$CARPETA/info_sistema_$DATA.txt"


{
    echo "Informació del sistema - $DATA"
    echo "Hostname: $(hostname)"
    echo "IP(s) assignada(s): $(ip a)"
    echo "Usuaris connectats actualment:"
    who
    echo "Target actual: $(systemctl get-default)"
    echo "Espai de disc disponible:"
    df -h
} > "$FITXER"


echo "Fitxer creat: $FITXER"

export DISPLAY=:0
export XAUTHORITY=/home/client1/.Xauthority
notify-send "ALERTA" "Aquesta màquina ha estat hackejada per Pellisa!"
```

He programat un script que recull informació del sistema i envia una notificació. El script:

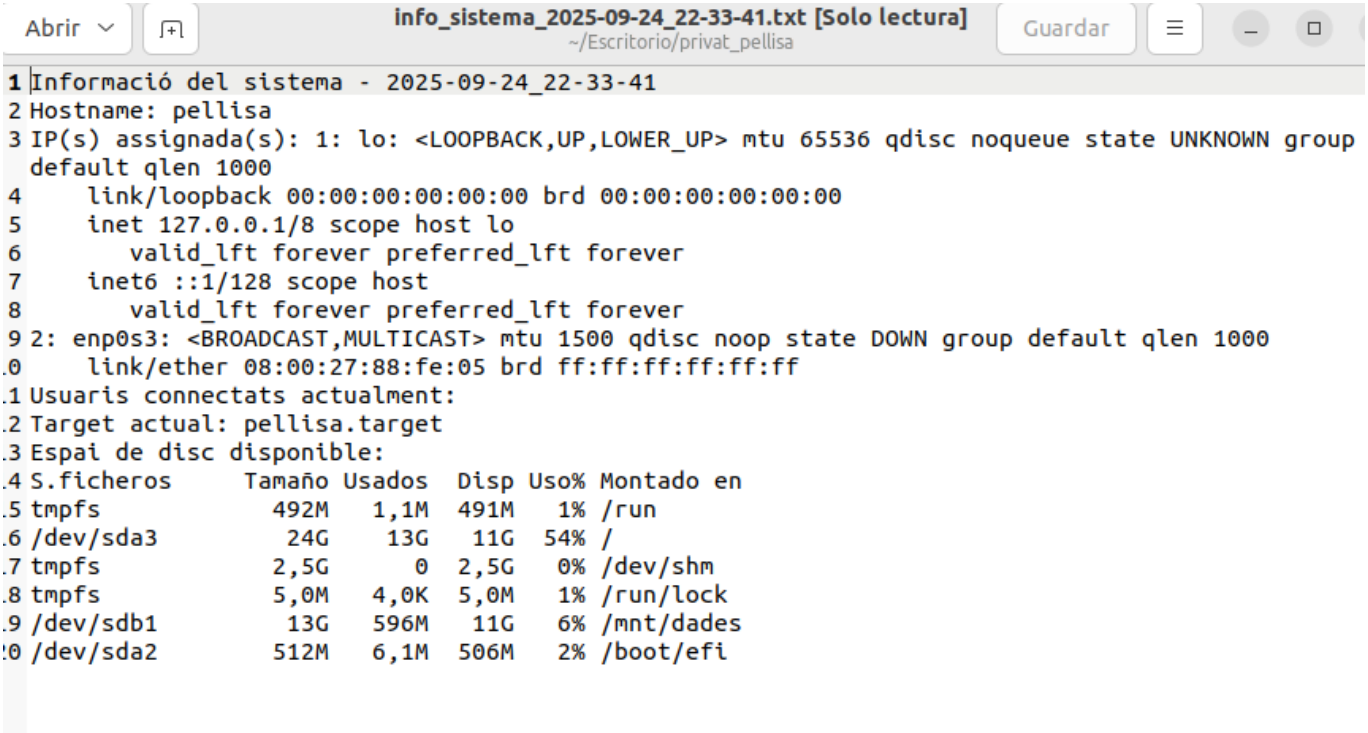
1. Crea una carpeta segura i un fitxer amb data/hora
2. Captura dades del sistema (hostname, IP, usuaris, espai de disc)
3. Guarda tot en un fitxer d'informe
4. Envia una notificació d'escriptori amb missatge "hackejada" (és broma, clar!)

```
client1@pellisa:~$ sudo chmod +x /usr/local/bin/script-pellisa.sh
```

He donat permisos d'execució al script perquè systemd pugui executar-lo quan el servei s'iniciï. Sense aquest permís, el servei fallaria al intentar executar el script.



Aqui ja provem de reinciar l'equip i veiem que ja s'ens carrega el target amb el servei i executa el script, a l'ultima part veiem que ens surt failed ja que, la notificació pop-up, ens genera un error.



I un cop reinciada veiem que s'ens ha generat l'informe amb les dades sensibles de l'equip.

MILLORES DEl SCRIPT

```
#!/bin/bash


set -euo pipefail

# ----------------------------
# Configuració
# ----------------------------
CARPETA="/home/client1/Escritorio/privat_pellisa"
CAPTURES_DIR="$CARPETA/captures"
MSMTP_CONF="/home/client1/.msmtprc"    # fitxer de configuració msmtp
DESTINATARI="josepmariapellisa@iesebre.com"
ASSUMPTE_PREFIX="Informe+Captura -"
WAIT_SECONDS=5

mkdir -p "$CAPTURES_DIR"
chmod 700 "$CARPETA" 2>/dev/null || true
chmod 700 "$CAPTURES_DIR" 2>/dev/null || true
```

Aquest script de bash està pensat per fer captures de pantalla automàtiques i enviar-les per correu electrònic. Primer defineix algunes variables importants com la carpeta de treball "/home/client1/Excritorio/privat_pellisa", la carpeta on es guardaran les captures, el fitxer de configuració del msmtp (que és un client de correu), l'adreça de correu del destinatari "josepmariapellisa@iesebre.com", el prefix de l'assumpte del correu i el temps d'espera entre captures (5 segons). Després crea la carpeta de captures si no existeix i li assigna permisos restringits (700) tant a la carpeta principal com a la de captures, amb un "2>/dev/null || true" perquè no mostri errors si no pot canviar els permisos. L'script està preparat per capturar pantalles periòdicament i enviar-les automàticament per correu al destinatari especificat.

```
# ----------------------------
# Noms de fitxer amb timestamp
# ----------------------------
DATA=$(date '+%Y-%m-%d_%H-%M-%S')
FITXER="$CARPETA/info_sistema_$DATA.txt"
ASSUMPTE="$ASSUMPTE_PREFIX $DATA"

SCREENSHOT_PNG="$CAPTURES_DIR/screenshot_$DATA.png"
PROCS_TXT="$CAPTURES_DIR/processos_$DATA.txt"
WM_TXT="$CAPTURES_DIR/finestres_$DATA.txt"
TERM_TXT="$CAPTURES_DIR/terminal_$DATA.txt"
TAR_CAPTURES="$CAPTURES_DIR/captures_$DATA.tar.gz"

# Boundary segur per MIME (no comenci amb -)
BOUNDARY="====$(date +%s)_$$===="
```

En aquesta part del script definim els noms dels fitxers que utilitzarem, afegint-hi un timestamp perquè cada execució tingui noms únics. Creem una variable DATA amb la data i hora actual en format any-mes-dia_hora-minut-segon. Definim FITXER com a l'arxiu d'informació del sistema amb aquesta data, i ASSUMPTE com l'assumpte del correu amb el prefix i la data. També configurem els noms per a la captura de pantalla en

PNG, els arxius de text per als processos actius, les finestres obertes i la informació del terminal, a més d'un arxiu TAR comprimit que agruparà totes les captures. Finalment, establim un BOUNDARY únic per a l'estructura MIME del correu electrònic, assegurant-nos que no comenci amb guió per evitar conflictes.

```
# ---------------------------
# Funcions auxiliars
# ---------------------------

take_screenshot() {
  # X11
  if [ -n "${DISPLAY:-}" ]; then
    if command -v scrot &> /dev/null; then
      scrot "$SCREENSHOT_PNG" 2>/dev/null && return 0
    elif command -v import &> /dev/null; then
      import -window root "$SCREENSHOT_PNG" 2>/dev/null && return 0
    fi
  fi

  return 1
}


wait_for_graphical() {
  local tries=0
  local maxtries=10
  while [ $tries -lt $maxtries ]; do
    if [ -n "${DISPLAY:-}" ] || [ -n "${WAYLAND_DISPLAY:-}" ]; then
      echo "Sessió gràfica detectada (DISPLAY/WAYLAND). Esperant ${WAIT_SECONDS}s addicionals..." >&2
      sleep "$WAIT_SECONDS"
      return 0
    fi
    tries=$((tries+1))
    sleep 1
  done
  echo "No s'ha detectat sessió gràfica després de ${maxtries}s; esperant ${WAIT_SECONDS}s abans d'intentar captura igualment..." >&2
  sleep "$WAIT_SECONDS"
  return 1
}
```

En aquesta secció realitzem les captures del sistema. Primer esperem que l'entorn gràfic estigui disponible utilitzant la funció wait_for_graphical. A continuació, intentem fer una captura de pantalla mitjançant la funció take_screenshot. Si la captura s'executa correctament, mostrem un missatge indicant on s'ha guardat el fitxer PNG. En cas contrari, si no es pot realitzar la captura (ja sigui per falta d'entorn gràfic o perquè falten les eines necessàries), informem de l'error i deixem la variable SCREENSHOT_PNG buida per indicar que no disposem d'aquesta captura.

```
# ---------------------------
# Captures: espera, pantalla, processos, finestres, historial
# ---------------------------
wait_for_graphical

# 1) Captura de pantalla
if take_screenshot; then
  echo "✅ Captura de pantalla guardada en: $SCREENSHOT_PNG"
else
  echo "⚠️ No s'ha pogut fer captura de pantalla (no hi ha entorn gràfic o manquen eines)."
  SCREENSHOT_PNG=""
fi
```

En aquesta secció executem les diferents captures del sistema. Primer esperem que l'entorn gràfic estigui disponible cridant la funció wait_for_graphical. A continuació, intentem realitzar una captura de pantalla mitjançant la funció take_screenshot. Si la captura s'executa amb èxit, mostrem un missatge per consola indicant la ruta on s'ha desat el fitxer PNG. En cas contrari, si no es pot completar la captura (ja sigui per absència d'entorn gràfic o per manca de les eines necessàries), notifiquem l'error i establim la variable SCREENSHOT_PNC com a buida per indicar que no tenim aquesta captura disponible.

```
# 2) Processos i arbre
{
  echo "== ps aux (arbre, fins 200 línies) =="
  ps aux --forest | head -n 200 || ps aux | head -n 200
  echo
  echo "== Top 20 per ús CPU =="
  ps -eo pid,uid,cmd,%cpu,%mem --sort=-%cpu | head -n 20
} > "$PROCS_TXT" 2>/dev/null || true
chmod 600 "$PROCS_TXT" 2>/dev/null || true
```

En aquesta part capturem informació dels processos del sistema. Executem una sèrie de comandos que redirigim cap a l'arxiu de processos: primer mostrem l'arbre de processos amb "ps aux --forest" limitat a 200 línies, i si falla mostrem la llista simple de processos. Després generem un llistat dels 20 processos que consumeixen més CPU, mostrant el PID, l'usuari, la comanda i el percentatge d'ús de CPU i memòria. Tota aquesta informació es guarda en el fitxer de processos definit anteriorment, ignorant possibles errors amb "2>/dev/null || true". Finalment, assignem permisos restrictius (600) a aquest fitxer per garantir la seva seguretat.

```
# 3) Finestres obertes (wmctrl o xdotool)
if command -v wmctrl &> /dev/null; then
  wmctrl -lp > "$WM_TXT" 2>/dev/null || true
elif command -v xdotool &> /dev/null; then
  xdotool search --onlyvisible --name . 2>/dev/null | while read -r id; do
    xdotool getwindowname "$id"
  done > "$WM_TXT" 2>/dev/null || true
else
  printf "No hi ha eina de finestres disponible (wmctrl/xdotool)\n" > "$WM_TXT"
fi
chmod 600 "$WM_TXT" 2>/dev/null || true
```

En aquesta secció capturem informació sobre les finestres obertes en l'entorn gràfic. Primer comprovem si tenim disponible l'eina "wmctrl" al sistema, i en cas afirmatiu, executem "wmctrl -lp" per obtenir una llista detallada de les finestres amb els seus identificadors i processos. Si no disposem de "wmctrl", llavors cerquem l'eina "xdotool" i l'utilitzem per buscar totes les finestres visibles i obtenir els seus noms. En cas que no trobem cap d'aquestes dues eines, escrivim un missatge d'error en el fitxer indicant que no hi ha eina de gestió de finestres disponible. Finalment, assignem permisos restrictius (600) al fitxer de finestres per garantir la seva seguretat.

```
# 4) Historial de shell (si existeix)
if [ -f "$HOME/.bash_history" ]; then
  printf "== Últim historial bash (últimes 200 línies) ==\n" > "$TERM_TXT"
  tail -n 200 "$HOME/.bash_history" >> "$TERM_TXT" 2>/dev/null || true
else
  printf "No s'ha trobat .bash_history o permisos insuficients\n" > "$TERM_TXT"
fi
chmod 600 "$TERM_TXT" 2>/dev/null || true

# 5) Empaquetar captures en un tar.gz (creem una temp dir per evitar problemes de rutes)
TMPDIR="$(mktemp -d "$CAPTURES_DIR/tmp_XXXX")"
trap 'rm -rf "$TMPDIR"' EXIT
```

En aquesta secció capturem l'historial de la shell i preparem l'empaquetament de les captures. Primer comprovem si existeix el fitxer .bash_history a la carpeta de l'usuari. Si existeix, n'extraiem les últimes 200 línies amb el comandament tail i les guardem al fitxer d'informació del terminal. En cas que no es trobi el

fitxer d'historial o no hi hagi permisos suficients, escrivim un missatge d'error al fitxer. Després assignem permisos restrictius (600) al fitxer de terminal per seguretat. A continuació, creem un directori temporal amb un nom únic dins de la carpeta de captures utilitzant mktemp, i configurem un trap per assegurar-nos que aquest directori temporal s'eliminarà automàticament quan l'script finalitzi, independentment de com acabi.

```
# Copiem els fitxers que existisquen al tmpdir
cp --preserve=mode,links "$FITXER" "$TMPDIR/" 2>/dev/null || cp "$FITXER" "$TMPDIR/"
[ -n "${SCREENSHOT_PNG:-}" ] && [ -f "$SCREENSHOT_PNG" ] && cp --preserve=mode,links "$SCREENSHOT_PNG" "$TMPDIR/" 2>/dev/null || true
[ -f "$PROCS_TXT" ] && cp --preserve=mode,links "$PROCS_TXT" "$TMPDIR/" 2>/dev/null || true
[ -f "$WM_TXT" ] && cp --preserve=mode,links "$WM_TXT" "$TMPDIR/" 2>/dev/null || true
[ -f "$TERM_TXT" ] && cp --preserve=mode,links "$TERM_TXT" "$TMPDIR/" 2>/dev/null || true
```

En aquesta secció copiem tots els fitxers de captures existents al directori temporal que hem creat. Primer intentem copiar el fitxer principal d'informació del sistema preservant els permisos i enllaços. Si aquesta còpia falla, fem una còpia normal. Després, només si la variable SCREENSHOT_PNG no està buida i el fitxer existeix realment, copiem la captura de pantalla al directori temporal. Seguidament, comprovem l'existència dels fitxers de processos, finestres i terminal, i per a cadascun d'ells que existeixi, el copiem al directori temporal preservant els seus permisos i enllaços. Totes aquestes operacions inclouen redireccions d'error per evitar que possibles fallades en la còpia interrompin l'execució de l'script.

```
# Create tar.gz from tmpdir contents
if find "$TMPDIR" -mindepth 1 | read -r _; then
  tar -C "$TMPDIR" -czf "$TAR_CAPTURES" . 2>/dev/null
  chmod 600 "$TAR_CAPTURES" 2>/dev/null || true
  echo "✅ Captures empaquetades a: $TAR_CAPTURES"
else
  echo "⚠️ No hi ha fitxers per empaquetar."
  TAR_CAPTURES=""
fi
```

En aquesta secció creem l'arxiu comprimit amb totes les captures. Primer comprovem si el directori temporal conté algun fitxer utilitzant la comanda find combinada amb read. Si hi ha contingut al directori temporal, executem la comanda tar per crear un arxiu comprimit en format .tar.gz que inclogui tots els fitxers del directori temporal. Després assignem permisos restrictius (600) a l'arxiu comprimit per garantir la seva seguretat i mostrem un missatge per consola indicant on s'ha guardat l'arxiu. Si el directori temporal està buit, mostrem un missatge indicant que no hi ha fitxers per empaquetar i deixem la variable TAR_CAPTURES buida per indicar que no s'ha generat cap arxiu.

```
# Creacio tar.gz desde el contigut de tmpdir
if find "$TMPDIR" -mindepth 1 | read -r _; then
  tar -C "$TMPDIR" -czf "$TAR_CAPTURES" . 2>/dev/null
  chmod 600 "$TAR_CAPTURES" 2>/dev/null || true
  echo "✅ Captures empaquetades a: $TAR_CAPTURES"
else
  echo "⚠️ No hi ha fitxers per empaquetar."
  TAR_CAPTURES=""
fi
```

En aquesta secció creem l'arxiu comprimit amb tot el contingut capturat. Primer comprovem si el directori temporal conté algun fitxer utilitzant la comanda `find` juntament amb `read` per detectar si hi ha elements dins del directori. Si trobem fitxers, executem la comanda `tar` per crear un arxiu comprimit en format .tar.gz que contingui tots els fitxers del directori temporal, canviant prèviament al directori amb l'opció `-C` per

evitar incloure rutes completes. Després assignem permisos restrictius (600) a l'arxiu comprimit per motius de seguretat i mostrem un missatge d'èxit per consola. Si el directori temporal està buit, indiquem que no hi ha fitxers per empaquetar i deixem la variable TAR_CAPTURES buida per reflectir aquest estat.

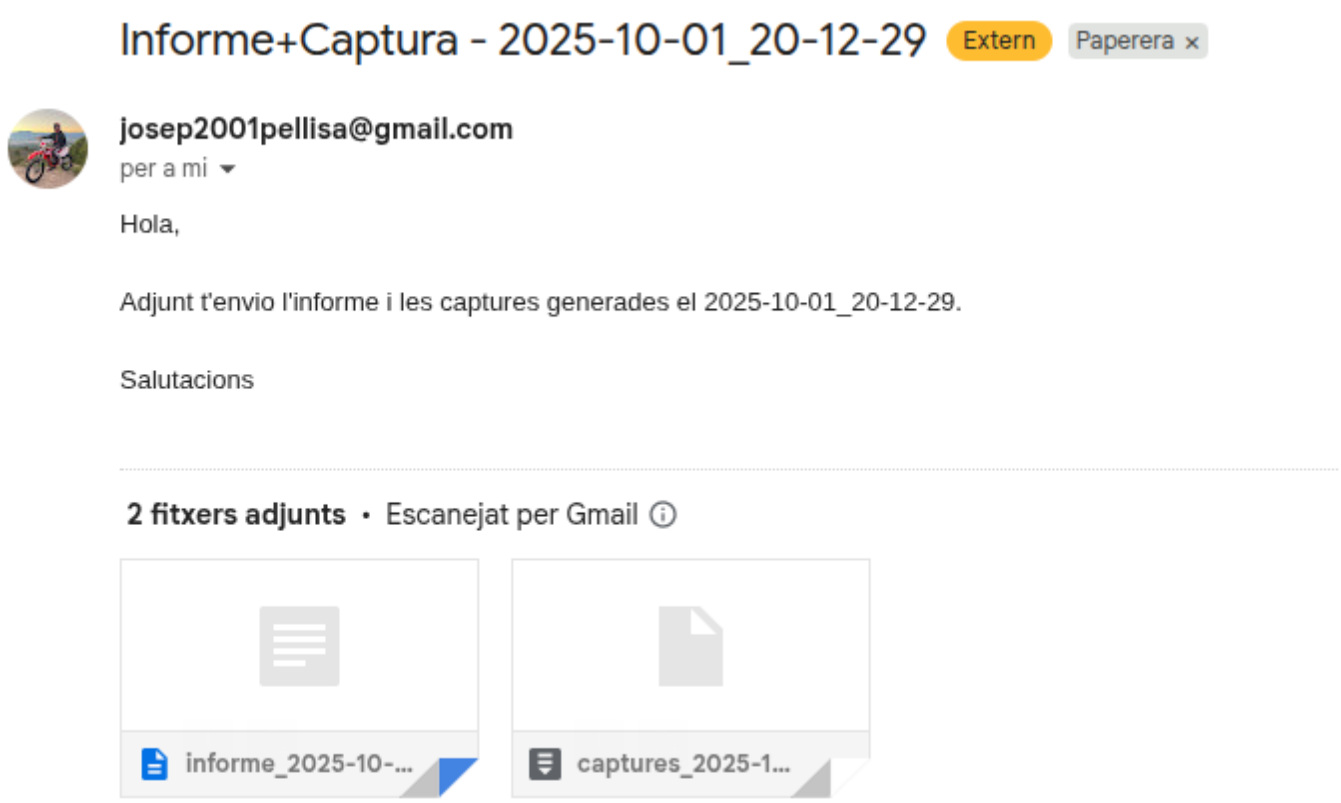
```
# ----------------------------
# Enviament per msmtp amb adjunt (MIME multipart)
# ----------------------------
if ! command -v msmtp &> /dev/null; then
  echo "ERROR: msmtp no està instal·lat. Instal·la'l: sudo apt install msmtp"
  exit 1
fi
```

En aquesta secció preparem l'enviament del correu electrònic amb les captures. Primer comprovem si tenim instal·lat el client de correu `msmtp` al sistema. Si no el trobem disponible, mostrem un missatge d'ERROR per consola indicant que `msmtp` no està instal·lat i proporcionem la comanda necessària per instal·lar-lo (`sudo apt install msmtp`). A continuació, finalitzem l'execució de l'script amb codi d'error 1 per evitar que continuï l'enviament sense tenir l'eina necessària. Aquesta comprovació és crucial per assegurar-nos que disposem de tot el necessari abans d'intentar enviar el correu electrònic amb les captures recopilades.

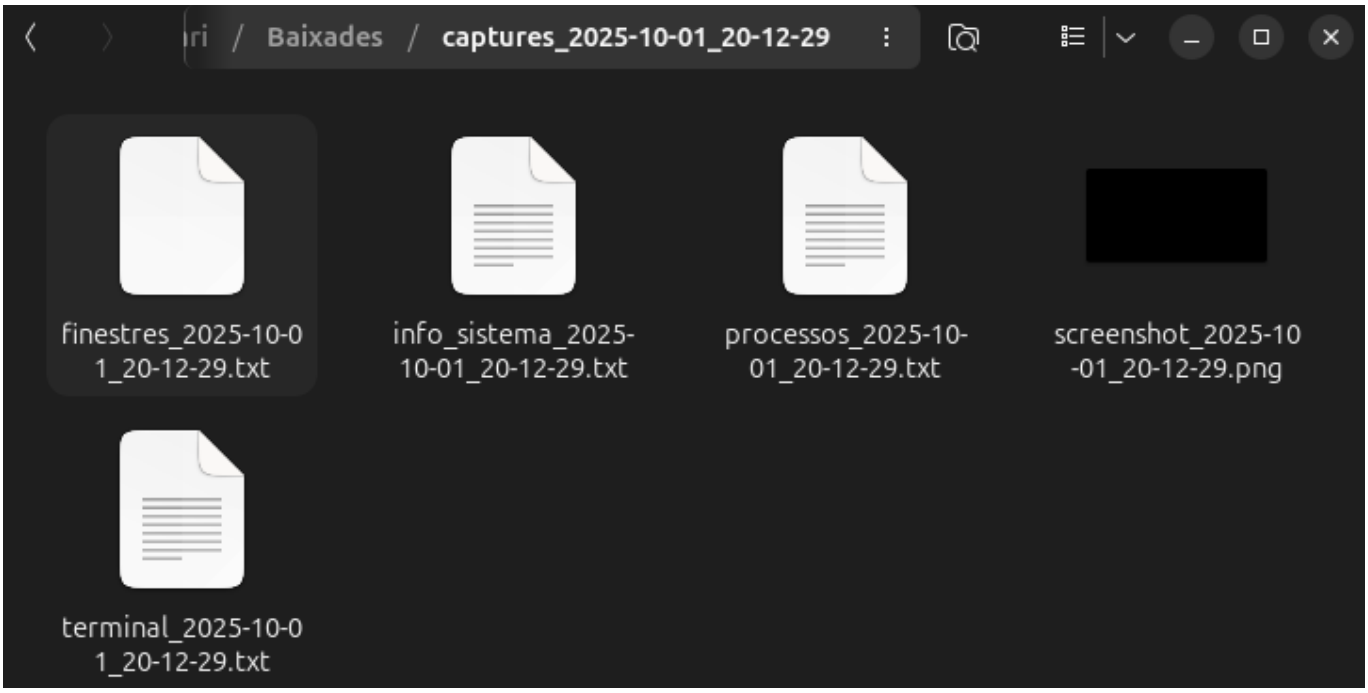
```
# Construïm missatge MIME amb adjunt si existeix
{
  printf "To: %s\n" "$DESTINATARI"
  printf "Subject: %s\n" "$ASSUMPTE"
  printf "MIME-Version: 1.0\n"
  if [ -n "${TAR_CAPTURES:-}" ] && [ -f "$TAR_CAPTURES" ]; then
    printf -- "Content-Type: multipart/mixed; boundary=\"%s\"\n\n" "$BOUNDARY"
    printf -- "--%s\n" "$BOUNDARY"
    printf -- "Content-Type: text/plain; charset=UTF-8\n"
    printf -- "Content-Transfer-Encoding: 7bit\n\n"
    printf -- "Hola,\n\nAdjunt t'envio l'informe i les captures generades el %s.\n\nSalutacions\n\n" "$DATA"

    printf -- "--%s\n" "$BOUNDARY"
    printf -- "Content-Type: text/plain; name=\"informe_%s.txt\"\n" "$DATA"
    printf -- "Content-Transfer-Encoding: 7bit\n"
    printf -- "Content-Disposition: inline; filename=\"informe_%s.txt\"\n\n" "$DATA"
    cat "$FITXER"
    printf -- "\n--%s\n" "$BOUNDARY"

    printf -- "Content-Type: application/gzip; name=\"%s\"\n" "$(basename "$TAR_CAPTURES")"
    printf -- "Content-Transfer-Encoding: base64\n"
    printf -- "Content-Disposition: attachment; filename=\"%s\"\n\n" "$(basename "$TAR_CAPTURES")"
    base64 "$TAR_CAPTURES"
    printf -- "\n--%s--\n" "$BOUNDARY"
  else
    printf -- "Content-Type: text/plain; charset=UTF-8\n"
    printf -- "Content-Transfer-Encoding: 7bit\n\n"
    printf -- "Hola,\n\nAdjunt (inline) l'informe generat el %s.\n\n" "$DATA"
    cat "$FITXER"
  fi
} | msmtp -C "$MSMTP_CONF" -a gmail "$DESTINATARI"

if [ "$?" -eq 0 ]; then
  echo "📧 Correu enviat correctament a $DESTINATARI"
else
  echo "❌ Error enviant el correu amb msmtp. Comprova /home/client1/.msmtprc i la connexió."
  exit 1
fi

# Neteja (el trap eliminarà tmpdir)
exit 0
```

En aquesta secció final construïm i enviem el correu electrònic amb les captures. Primer generem la capçalera del missatge MIME amb les dades del destinatari, assumpte i versió. Si existeix l'arxiu comprimit amb les captures, creem un missatge multipart que inclou tant el text del correu com l'arxiu adjunt codificat en base64. Estructurem el missatge amb boundaries per separar les diferents parts. Si no hi ha arxiu per

adjuntar, enviem només el text pla amb la informació del sistema. El missatge complet es redirigeix a msmtp per al seu enviament, utilitzant el fitxer de configuració especificat i el compte de Gmail. Verifiquem l'estat de l'enviament: si és exitós, ho confirmem per consola; si hi ha error, informem i sortim amb codi d'error. Finalment, netegem els recursos temporals (gràcies al trap configurat anteriorment) i sortim amb èxit.



Aquest és un exemple de com es veuria el correu electrònic que s'envia amb l'script. El missatge inclou l'assumpte amb data i hora exacta "Informe+Captura - 2025-10-01_20-12-29", està adreçat a l'usuari Josep2001pellisa@gmail.com i conté un text amable sol·licitant que es revisin les captures adjuntes. El cos del missatge és breu i directe, amb una salutació i la informació de la data de generació. S'adjunten dos fitxers: l'informe i les captures comprimides, amb les dates corresponents.



Aquests són els fitxers individuals que es generen amb cada execució de l'script de captures. Cada fitxer inclou la data i hora exacta (2025-10-01_20-12-29) per garantir que siguin únics i es puguin distingir entre diferents execucions. Tenim cinc tipus de fitxers: "finestres" que conté informació sobre les finestres obertes en l'entorn gràfic, "info_sistema" amb dades generals del sistema, "processos" que mostra els processos actius i l'ús de recursos, "screenshot" que és la captura de pantalla en format PNG, i "terminal" amb l'historial de comandes de la shell. Tots aquests fitxers s'empaqueten després en un sol arxiu comprimit .tar.gz per al seu enviament per correu electrònic.

```
========================================
   INFORME DEL SISTEMA - 2025-10-01_20-12-29
========================================

Host / Data:
  Hostname: pellisa
  Data/hora: Wed, 01 Oct 2025 20:12:29 +0200

IP(s) IPv4 actives:
    - 192.168.203.156/24 (enp0s3)

Usuaris connectats actualment:
  client1  tty2         2025-10-01 19:41 (tty2)

Target actual: pellisa.target

Espai de disc (df -h):
  S.ficheros      Tamaño Usados  Disp Uso% Montado en
  tmpfs             492M   1,5M  491M   1% /run
  /dev/sda3          24G    13G  9,9G  57% /
  tmpfs             2,5G    12K  2,5G   1% /dev/shm
  tmpfs             5,0M   4,0K  5,0M   1% /run/lock
  /dev/sdb1          13G   596M   11G   6% /mnt/dades
  /dev/sda2         512M   6,1M  506M   2% /boot/efi
  tmpfs             492M   116K  492M   1% /run/user/1000
  /dev/sr0           51M    51M     0 100% /media/client1/VBox_GAs_7.2.2

========================================
```

Aquest és el contingut de l'arxiu `info_sistema_2025-10-01_20-12-29.txt` que genera l'script. Conté un informe complet de l'estat del sistema en el moment de les captures. Inclou informació bàsica com el nom del host ("pellisa"), la data i hora exacta de la captura, les adreces IP actives (en aquest cas 192.168.203.156), els usuaris connectats (client1 en tty2), el target actual del systemd i un resum de l'ús dels discs amb `df -h`. Es pot observar que el disc principal (/dev/sda3) està al 57% de la seva capacitat, hi ha un disc de dades addicional (/dev/sdb1) muntat al 6%, i s'ha detectat un CD-ROM virtual de VirtualBox muntat al 100%. Aquesta informació és valuosa per fer un diagnòstic ràpid de l'estat del sistema.

```
== ps aux (arbre, fins 200 línies) ==
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           2  0.0  0.0      0     0 ?        S    19:41   0:00 [kthreadd]
root           3  0.0  0.0      0     0 ?        S    19:41   0:00  \_ [pool_workqueue_release]
root           4  0.0  0.0      0     0 ?        I<   19:41   0:00  \_ [kworker/R-rcu_g]
root           5  0.0  0.0      0     0 ?        I<   19:41   0:00  \_ [kworker/R-rcu_p]
root           6  0.0  0.0      0     0 ?        I<   19:41   0:00  \_ [kworker/R-slub_]
root           7  0.0  0.0      0     0 ?        I<   19:41   0:00  \_ [kworker/R-netns]
root          10  0.0  0.0      0     0 ?        I<   19:41   0:00  \_ [kworker/0:0H-events_highpri]
root          11  0.0  0.0      0     0 ?        I    19:41   0:00  \_ [kworker/u4:0-ext4-rsv-conversion]
root          12  0.0  0.0      0     0 ?        I<   19:41   0:00  \_ [kworker/R-mm_pe]
root          13  0.0  0.0      0     0 ?        I    19:41   0:00  \_ [rcu_tasks_kthread]
root          14  0.0  0.0      0     0 ?        I    19:41   0:00  \_ [rcu_tasks_rude_kthread]
root          15  0.0  0.0      0     0 ?        I    19:41   0:00  \_ [rcu_tasks_trace_kthread]
root          16  0.0  0.0      0     0 ?        S    19:41   0:00  \_ [ksoftirqd/0]
root          17  0.0  0.0      0     0 ?        I    19:41   0:01  \_ [rcu_preempt]
root          18  0.0  0.0      0     0 ?        S    19:41   0:00  \_ [migration/0]
root          19  0.0  0.0      0     0 ?        S    19:41   0:00  \_ [idle_inject/0]
```

Aquest és el contingut de l'arxiu de processos que genera l'script. Mostra una llista dels processos actius en el sistema en el moment de la captura, limitada a les primeres 200 línies. La taula inclou informació detallada de cada procés: l'usuari que l'executa, el PID (Identificador de Procés), el percentatge d'ús de CPU i memòria, la memòria virtual i resident utilitzada, la terminal associada, l'estat del procés, l'hora d'inici, el temps d'execució i la comanda que l'ha llançat.

```
== Últim historial bash (últimes 200 línies) ==
clear
sudo su
appt isntall quota
sudo su
hostname pellisa
sudo hostname pellisa
usermod -l pellis client1
reboot
sudo nano /etc/systemd/system/pellisa.target
sudo mkdir -p /etc/systemd/system/pellisa.target.wants
sudo nano /usr/local/bin/script-pellisa.sh
sudo chmod +x /usr/local/bin/script-pellisa.sh
cd /usr/local/bin
ls
./script-pellisa.sh
sudo ./script-pellisa.sh
sudo cat /var/log/script-pellisa.log
sudo nano /etc/systemd/system/servei_pellisa.service
sudo su
sudo nano /usr/local/
ls /usr/local/
cd /usr/local/
ls
cd bin/
ls
sudo su
```

Finalment aquest és el contingut de l'arxiu d'historial del terminal que captura l'script. Mostra les últimes 200 comandes executades per l'usuari en la shell bash. Es pot veure un historial d'activitats d'administració del sistema que inclou: canvis en el nom del host (hostname pellisa), instal·lació de paquets (apt install quota), modificació d'usuaris (usermod), creació i configuració de serveis systemd, edició de scripts amb nano, i canvis de permisos d'execució.