HDD Tray ermitteln LED blinken und/oder austauschen

Beschreibung:

Es gibt ein tolles Tool. Mit der man die LED zum Tray blinken lassen kann.
Noch ein kleines Script dazu wo man nur dass device angeben muss.Ein weiteres script zum Datenträger tausch.
Lässt slot blinken.
festplatte tauschen
Alte festplatte wird deregistriert und neue hinzugefügt.
LED ausgeschaltet.
BAM

Durchführung:

```
apt-get install -y lsscsi sg3-utils
```

Festplatten anzeigen lassen:

```
lsscsi -g
```

Ausgabe:

```
[1:0:0:0]    cd/dvd  Slimtype DVD A  DS8ACSH    LP5M  /dev/sr0   /dev/sg17
[3:0:0:0]    disk    SEAGATE  XS400ME70004     0004  /dev/sda   /dev/sg0
[3:0:1:0]    disk    SEAGATE  ST20000NM002D    E004  /dev/sdb   /dev/sg1
[3:0:2:0]    disk    WDC      WUS721010AL5204  C980  /dev/sdc   /dev/sg2
[3:0:3:0]    disk    SEAGATE  ST20000NM002D    E006  /dev/sdd   /dev/sg3
[3:0:4:0]    disk    WDC      WUS721010AL5204  C980  /dev/sde   /dev/sg4
[3:0:5:0]    disk    SEAGATE  ST20000NM002D    E006  /dev/sdf   /dev/sg5
```

```
[3:0:6:0]    disk    WDC      WUS721010AL5204  C980  /dev/sdg   /dev/sg6
[3:0:7:0]    disk    WDC      WUS721010AL5204  C980  /dev/sdh   /dev/sg7
[3:0:8:0]    disk    WDC      WUS721010AL5204  C980  /dev/sdi   /dev/sg8
[3:0:9:0]    disk    WDC      WUS721010AL5204  C980  /dev/sdj   /dev/sg9
[3:0:10:0]   disk    WDC      WUS721010AL5204  C980  /dev/sdk   /dev/sg10
[3:0:11:0]   disk    SEAGATE  XS400ME70004     0004  /dev/sdl   /dev/sg11
[3:0:12:0]   disk    WDC      WUS721010AL5204  C980  /dev/sdm   /dev/sg12
[3:0:13:0]   disk    WDC      WUS721010AL5204  C980  /dev/sdn   /dev/sg13
[3:0:14:0]   disk    WDC      WUS721010AL5204  C980  /dev/sdo   /dev/sg14
[3:0:15:0]   disk    WDC      WUS721010AL5204  C980  /dev/sdp   /dev/sg15
[3:0:16:0]   enclosu BROADCOM VirtualSES       03    -          /dev/sg16
[4:0:0:0]    cd/dvd  AMI      Virtual CDROM0   1.00  /dev/sr1   /dev/sg18
```

Script:

```
nano /usr/local/bin/identify-disk.sh
```

Inhalt

```
#!/usr/bin/env bash
set -euo pipefail

usage() {
  cat <<USAGE
Usage:
  identify-disk [--on|--off|--blink SECONDS] [--symlink|--ses] [--all] [/dev/sdX|sdX|sdXn]

Optionen:
  --on / --off / --blink SECONDS   LED steuern
  --symlink                        nur sysfs/enclosure_device* nutzen
  --ses                            nur SES (sg_ses) nutzen
  --all                            auf alle verfügbaren Slots anwenden
  (ohne Methode)                   auto: erst symlink, dann SES-Fallback

Beispiele:
  identify-disk --on        sdi
  identify-disk --blink 10  sdg8
  identify-disk --off       --symlink sdi
  identify-disk --on        --ses     sdi
  identify-disk --off       --all             # alle LEDs aus
```

```
  identify-disk --blink 5   --all --symlink   # alle via sysfs blinken
USAGE
}

MODE="on"
BLINK_SECS=0
METHOD="auto"   # auto | symlink | ses
ALL=0
DISK=""

[[ $# -lt 1 ]] && { usage; exit 1; }

while [[ $# -gt 0 ]]; do
  case "${1:-}" in
    --on) MODE="on"; shift ;;
    --off) MODE="off"; shift ;;
    --blink) MODE="blink"; BLINK_SECS="${2:-}"; [[ -z "$BLINK_SECS" ]] && { echo "Fehlende Sekunden bei --
blink"; exit 1; }; shift 2 ;;
    --symlink) METHOD="symlink"; shift ;;
    --ses) METHOD="ses"; shift ;;
    --all) ALL=1; shift ;;
    -h|--help) usage; exit 0 ;;
    *)
      DISK="$1"; shift ;;
  esac
done

# ----------------------------
# Helpers (norm/req/printing)
# ----------------------------
need() { command -v "$1" >/dev/null 2>&1 || { echo "$1 fehlt. Bitte installieren."; exit 1; }; }

normalize_disk() {
  local d="$1"
  [[ "$d" =~ ^sd ]] && d="/dev/$d"
  local base="$(basename "$d")"
  local base_nopart="${base%%[0-9]*}"
  echo "/dev/$base_nopart"
}
```

```
# ---------------------------------------
# Einzel-Operationen (SYMLINK / per Disk)
# ---------------------------------------
do_symlink_one() {
  local disk="$1"
  local base_nopart="$(basename "$disk")"
  local p="/sys/class/block/${base_nopart}/device"
  local enc=( "$p"/enclosure_device* )
  [[ -e "${enc[0]}" ]] || { echo "Kein enclosure_device* für $disk gefunden."; return 2; }
  local LOCATE="${enc[0]}/locate"
  [[ -w "$LOCATE" ]] || { echo "'locate' nicht verfügbar/schreibbar unter ${enc[0]}."; return 3; }
  local slot="$(cat "${enc[0]}/slot" 2>/dev/null || echo "")"
  local slot_lbl=""; [[ -n "$slot" ]] && slot_lbl=" (Slot $slot)"
  case "$MODE" in
    on)    echo 1 | sudo tee "$LOCATE" >/dev/null; echo "ON  via sysfs$slot_lbl – $disk";;
    off)   echo 0 | sudo tee "$LOCATE" >/dev/null; echo "OFF via sysfs$slot_lbl – $disk";;
    blink) echo 1 | sudo tee "$LOCATE" >/dev/null; echo "ON  via sysfs$slot_lbl – $disk"; sleep "$BLINK_SECS";
echo 0 | sudo tee "$LOCATE" >/dev/null; echo "OFF via sysfs$slot_lbl – $disk";;
  esac
}

# ---------------------------------------
# Einzel-Operationen (SES / per Disk)
# ---------------------------------------
do_ses_one() {
  local disk="$1"
  need lsscsi; need sg_ses
  local LINE
  LINE="$(lsscsi -g | awk -v d="$disk" '{if ($(NF-1)==d) {print; exit}}')"
  [[ -n "$LINE" ]] || { echo "Konnte $disk nicht in 'lsscsi -g' finden."; return 6; }

  local HOST BUS TARGET LUN
  if [[ "$LINE" =~ \[([0-9]+):([0-9]+):([0-9]+):([0-9]+)\] ]]; then
    HOST="${BASH_REMATCH[1]}"; BUS="${BASH_REMATCH[2]}"; TARGET="${BASH_REMATCH[3]}";
LUN="${BASH_REMATCH[4]}"
  else
    echo "Konnte SCSI-Adresse nicht parsen."; return 7
  fi

  local SES_LINE SES_DEV
```

```
  SES_LINE="$(lsscsi -g | awk -v h="$HOST" -v b="$BUS" '$2=="enclosu" && $1 ~ ("\\["h":"b":"){print; exit}')"
  [[ -n "$SES_LINE" ]] || { echo "Kein SES/Enclosure auf ${HOST}:${BUS} gefunden (HW-RAID? Hersteller-CLI
nötig)."; return 8; }
  SES_DEV="$(echo "$SES_LINE" | awk '{print $NF}')"
  [[ -e "$SES_DEV" ]] || { echo "SES-Gerät existiert nicht: $SES_DEV"; return 9; }

  local SAS_PATH="/sys/class/scsi_device/${HOST}:${BUS}:${TARGET}:${LUN}/device/sas_address"
  [[ -r "$SAS_PATH" ]] || { echo "SAS-Adresse nicht lesbar: $SAS_PATH"; return 10; }
  local DISK_SAS; DISK_SAS="$(tr -d '\n' < "$SAS_PATH")"
  [[ -n "$DISK_SAS" ]] || { echo "Leere SAS-Adresse."; return 11; }

  local INDEX SLOTNUM
  read INDEX SLOTNUM < <(sg_ses -p aes "$SES_DEV" 2>/dev/null \
    | awk -v want="$DISK_SAS" '
        /Element index:/             {idx=$3; gsub(":","",idx); slot=""}
        /device slot number:/        {slot=$5}
        /SAS address:/ && $3 ~ /^0x/  {sas=$3; if (tolower(sas)==tolower(want)) {print idx, slot; exit}}
      ')
  [[ -n "${INDEX:-}" ]] || { echo "Konnte SES-Index für $disk nicht ermitteln."; return 12; }

  local slot_lbl=""; [[ -n "${SLOTNUM:-}" ]] && slot_lbl=" (Slot $SLOTNUM, idx $INDEX)" || slot_lbl=" (idx
$INDEX)"
  case "$MODE" in
    on)    sg_ses --index="$INDEX" --set=ident "$SES_DEV"; echo "ON  via SES$slot_lbl – $disk";;
    off)   sg_ses --index="$INDEX" --clear=ident "$SES_DEV"; echo "OFF via SES$slot_lbl – $disk";;
    blink) sg_ses --index="$INDEX" --set=ident "$SES_DEV"; echo "ON  via SES$slot_lbl – $disk"; sleep
"$BLINK_SECS"; sg_ses --index="$INDEX" --clear=ident "$SES_DEV"; echo "OFF via SES$slot_lbl – $disk";;
  esac
}

# ---------------------------------------
# ALL-Operationen (SYMLINK)
# ---------------------------------------
do_symlink_all() {
  local disks=()
  for d in /sys/class/block/sd*; do
    [[ -e "$d" ]] || continue
    local name="$(basename "$d")"
    [[ "$name" =~ [0-9]$ ]] && continue   # nur Whole Disks
    comp=( "$d"/device/enclosure_device* )
```

```
    [[ -e "${comp[0]}" ]] || continue
    disks+=( "/dev/$name" )
  done
  local n="${#disks[@]}"
  [[ $n -gt 0 ]] || { echo "Keine sysfs/enclosure Slots gefunden."; return 2; }

  if [[ "$MODE" == "blink" ]]; then
    # Alle an, warten, alle aus
    for disk in "${disks[@]}"; do
      local loc=( "/sys/class/block/$(basename "$disk")/device"/enclosure_device*/locate )
      [[ -w "${loc[0]}" ]] || continue
      echo 1 | sudo tee "${loc[0]}" >/dev/null
      local slot=""; [[ -r "${loc[0]%/*}/slot" ]] && slot="$(cat "${loc[0]%/*}/slot")"
      echo "ON  via sysfs$( [[ -n "$slot" ]] && echo " (Slot $slot)" ) – $disk"
    done
    sleep "$BLINK_SECS"
    for disk in "${disks[@]}"; do
      local loc=( "/sys/class/block/$(basename "$disk")/device"/enclosure_device*/locate )
      [[ -w "${loc[0]}" ]] || continue
      echo 0 | sudo tee "${loc[0]}" >/dev/null
      echo "OFF via sysfs – $disk"
    done
  else
    local val=$([[ "$MODE" == "on" ]] && echo 1 || echo 0)
    for disk in "${disks[@]}"; do
      local loc=( "/sys/class/block/$(basename "$disk")/device"/enclosure_device*/locate )
      [[ -w "${loc[0]}" ]] || continue
      echo $val | sudo tee "${loc[0]}" >/dev/null
      local slot=""; [[ -r "${loc[0]%/*}/slot" ]] && slot="$(cat "${loc[0]%/*}/slot")"
      echo "$( [[ $val -eq 1 ]] && echo ON || echo OFF ) via sysfs$( [[ -n "$slot" ]] && echo " (Slot $slot)" ) – $disk"
    done
  fi
}

# --------------------------------------
# ALL-Operationen (SES)
# --------------------------------------
do_ses_all() {
  need lsscsi; need sg_ses
  mapfile -t ENCS < <(lsscsi -g | awk '$2=="enclosu"{print $NF}')
```

```
  [[ ${#ENCS[@]} -gt 0 ]] || { echo "Kein SES/Enclosure gefunden."; return 3; }

  for SES_DEV in "${ENCS[@]}"; do
    # Alle belegten Element-Indices sammeln (die eine SAS address haben)
    mapfile -t IDX < <(sg_ses -p aes "$SES_DEV" 2>/dev/null \
      | awk '
          /Element index:/ {idx=$3; gsub(":","",idx); slot=""; has_sas=0}
          /device slot number:/ {slot=$5}
          /SAS address:/ && $3 ~ /^0x/ {has_sas=1}
          has_sas && /SAS address:/ {print idx "|" slot}
        ')
    [[ ${#IDX[@]} -gt 0 ]] || { echo "Keine befüllten Slots für $SES_DEV."; continue; }

    if [[ "$MODE" == "blink" ]]; then
      # alle an
      for pair in "${IDX[@]}"; do
        IFS='|' read -r index slot <<<"$pair"
        sg_ses --index="$index" --set=ident "$SES_DEV" || true
        [[ -n "$slot" ]] && echo "ON  via SES (Slot $slot, idx $index) – $SES_DEV" || echo "ON  via SES (idx $index) –
$SES_DEV"
      done
      sleep "$BLINK_SECS"
      # alle aus
      for pair in "${IDX[@]}"; do
        IFS='|' read -r index slot <<<"$pair"
        sg_ses --index="$index" --clear=ident "$SES_DEV" || true
        [[ -n "$slot" ]] && echo "OFF via SES (Slot $slot, idx $index) – $SES_DEV" || echo "OFF via SES (idx $index) –
$SES_DEV"
      done
    else
      local cmd=$([[ "$MODE" == "on" ]] && echo "--set=ident" || echo "--clear=ident")
      for pair in "${IDX[@]}"; do
        IFS='|' read -r index slot <<<"$pair"
        sg_ses --index="$index" $cmd "$SES_DEV" || true
        local state=$([[ "$MODE" == "on" ]] && echo ON || echo OFF)
        [[ -n "$slot" ]] && echo "$state via SES (Slot $slot, idx $index) – $SES_DEV" || echo "$state via SES (idx
$index) – $SES_DEV"
      done
    fi
  done
```

```
}

# ----------------------
# Main-Dispatch
# ----------------------
if [[ $ALL -eq 1 ]]; then
  case "$METHOD" in
    symlink)
      do_symlink_all
      ;;
    ses)
      do_ses_all
      ;;
    auto)
      # erst symlink-Variante probieren; wenn nichts gefunden, SES-all
      if ! do_symlink_all; then
        echo "(Auto) Fallback auf SES (alle)..."
        do_ses_all
      fi
      ;;
    *) echo "Unbekannte Methode: $METHOD"; usage; exit 1 ;;
  esac
  exit 0
fi

# Einzel-Disk-Pfad (keine --all)
[[ -n "$DISK" ]] || { echo "Bitte /dev/sdX angeben oder --all nutzen."; exit 1; }
DISK="$(normalize_disk "$DISK")"
[[ -b "$DISK" ]] || { echo "Gerät nicht gefunden: $DISK"; exit 1; }

case "$METHOD" in
  symlink)
    do_symlink_one "$DISK" || exit $?
    ;;
  ses)
    do_ses_one "$DISK" || exit $?
    ;;
  auto)
    if do_symlink_one "$DISK"; then
      exit 0
```

```
    else
      echo "(Auto) Fallback auf SES..."
      do_ses_one "$DISK"
    fi
    ;;
  *)
    echo "Unbekannte Methode: $METHOD"; usage; exit 1 ;;
esac
```

Ausführbar machen

```
chmod +x /usr/local/bin/identify-disk.sh
```

Hilfe/Bedienung

```
Usage:
  identify-disk [--on|--off|--blink SECONDS] [--symlink|--ses] [--all] [/dev/sdX|sdX|sdXn]

Optionen:
  --on / --off / --blink SECONDS    LED steuern
  --symlink                         nur sysfs/enclosure_device* nutzen
  --ses                             nur SES (sg_ses) nutzen
  --all                             auf alle verfügbaren Slots anwenden
  (ohne Methode)                    auto: erst symlink, dann SES-Fallback

Beispiele:
  identify-disk --on        sdi
  identify-disk --blink 10  sdg8
  identify-disk --off       --symlink sdi
  identify-disk --on        --ses     sdi
  identify-disk --off       --all             # alle LEDs aus
  identify-disk --blink 5   --all --symlink   # alle via sysfs blinken
```

Alte Laufwerke entfernen:

Sobald alte Laufwerke entfernt werden, bleiben Sie tot in der Liste.
Erst Laufwerk entfernen. Dann neu einschieben.
Das machen wir wieder mit einem script. Das Script wartet solange bis wir die festplatte getauscht haben.
wenn getauscht bestätigen.

```
nano /usr/local/bin/scsi-hotswap.sh
```

Inhalt

```
#!/usr/bin/env bash
set -euo pipefail

usage() {
  cat <<USAGE
Usage:
  scsi-hotswap [--method auto|symlink|ses] [--blink N|--no-led] [--no-prompt] /dev/sdX|sdX|sdXn

Optionen:
  --method M     M = auto (default) | symlink | ses
  --blink N      LED vor dem Tausch N Sekunden blinken (statt dauerhaft an)
  --no-led       keine LED-Steuerung
  --no-prompt    nicht auf ENTER warten (setze voraus, dass neue Platte schon steckt)

Beispiele:
  scsi-hotswap sdi
  scsi-hotswap --method symlink --blink 8 sdg8
  scsi-hotswap --method ses --no-prompt sdm
USAGE
}

METHOD="auto"      # auto | symlink | ses
BLINK_SECS=0       # 0 = steady on
USE_LED=1
PROMPT=1

[[ $# -lt 1 ]] && { usage; exit 1; }
while [[ $# -gt 0 ]]; do
  case "${1:-}" in
    --method) METHOD="${2:-}"; shift 2 ;;
    --blink)  BLINK_SECS="${2:-}"; [[ -z "$BLINK_SECS" ]] && { echo "Fehlende Sekunden bei --blink"; exit 1; };
shift 2 ;;
    --no-led) USE_LED=0; shift ;;
    --no-prompt) PROMPT=0; shift ;;
```

```
    -h|--help) usage; exit 0 ;;
    *) DISK="$1"; shift ;;
  esac
done

# --- Device normalisieren ---
[[ "$DISK" =~ ^sd ]] && DISK="/dev/$DISK"
BASE="$(basename "$DISK")"
BASE_NOPART="${BASE%%[0-9]*}"
DISK="/dev/$BASE_NOPART"
[[ -b "$DISK" ]] || { echo "Gerät nicht gefunden: $DISK"; exit 1; }

need() { command -v "$1" >/dev/null 2>&1 || { echo "$1 fehlt. Bitte installieren."; exit 1; }; }

# --------------------------
# LED via sysfs (symlink-Weg)
# --------------------------
# merkt sich LOCATE_PATH_CACHED+SLOT_CACHED, damit "off" nach dem Swap immer klappt
LOCATE_PATH_CACHED=""
SLOT_CACHED=""

led_symlink() {
  local mode="$1"  # on|off|blink
  local LOCATE_PATH=""
  local SLOT=""

  # bereits gecachten Pfad benutzen, wenn vorhanden
  if [[ -n "$LOCATE_PATH_CACHED" && -e "$LOCATE_PATH_CACHED" ]]; then
    LOCATE_PATH="$LOCATE_PATH_CACHED"
    SLOT="$SLOT_CACHED"
  else
    local p="/sys/class/block/${BASE_NOPART}/device"
    local enc=( "$p"/enclosure_device* )
    [[ -e "${enc[0]}" ]] || return 2
    LOCATE_PATH="${enc[0]}/locate"
    [[ -w "$LOCATE_PATH" ]] || return 3
    SLOT="$(cat "${enc[0]}/slot" 2>/dev/null || echo "")"
    LOCATE_PATH_CACHED="$LOCATE_PATH"
    SLOT_CACHED="$SLOT"
  fi
```

```
  local slot_label=""
  [[ -n "$SLOT" ]] && slot_label=" (Slot $SLOT)"

  case "$mode" in
    on)
      echo 1 | sudo tee "$LOCATE_PATH" >/dev/null
      echo "LED an${slot_label} via sysfs – $DISK"
      ;;
    off)
      echo 0 | sudo tee "$LOCATE_PATH" >/dev/null
      echo "LED aus${slot_label} via sysfs – $DISK"
      ;;
    blink)
      echo 1 | sudo tee "$LOCATE_PATH" >/dev/null
      echo "LED an${slot_label} – blinke ${BLINK_SECS}s…"
      sleep "$BLINK_SECS"
      echo 0 | sudo tee "$LOCATE_PATH" >/dev/null
      echo "LED aus."
      ;;
  esac
  return 0
}

# ---------------------------------------------
# SES-Index + Slotnummer + SES-Device ermitteln (per SAS)
# ---------------------------------------------
get_ses_index() {
  # Outputs (echo): "<SES_DEV> <INDEX> <SLOTNUM> <HOST> <BUS> <TARGET> <LUN>"
  need lsscsi; need sg_ses

  local LINE
  LINE="$(lsscsi -g | awk -v d="$DISK" '{if ($(NF-1)==d) {print; exit}}')"
  [[ -n "$LINE" ]] || { echo "Konnte $DISK nicht in 'lsscsi -g' finden." >&2; return 1; }

  if [[ "$LINE" =~ \[([0-9]+):([0-9]+):([0-9]+):([0-9]+)\] ]]; then
    local HOST="${BASH_REMATCH[1]}" BUS="${BASH_REMATCH[2]}" TARGET="${BASH_REMATCH[3]}"
LUN="${BASH_REMATCH[4]}"
  else
    echo "Konnte SCSI-Adresse nicht parsen." >&2; return 1
```

```
  fi

  local SES_LINE SES_DEV
  SES_LINE="$(lsscsi -g | awk -v h="$HOST" -v b="$BUS" '$2=="enclosu" && $1 ~ ("\\["h":"b":"){print; exit}')"
  [[ -n "$SES_LINE" ]] || { echo "Kein SES/Enclosure auf ${HOST}:${BUS} gefunden (HW-RAID?)." >&2; return 1;
}
  SES_DEV="$(echo "$SES_LINE" | awk '{print $NF}')"
  [[ -e "$SES_DEV" ]] || { echo "SES-Gerät existiert nicht: $SES_DEV" >&2; return 1; }

  local SAS_PATH="/sys/class/scsi_device/${HOST}:${BUS}:${TARGET}:${LUN}/device/sas_address"
  [[ -r "$SAS_PATH" ]] || { echo "SAS-Adresse nicht lesbar: $SAS_PATH" >&2; return 1; }
  local DISK_SAS; DISK_SAS="$(tr -d '\n' < "$SAS_PATH")"
  [[ -n "$DISK_SAS" ]] || { echo "Leere SAS-Adresse." >&2; return 1; }

  # In einem Durchlauf: Index + physische Slotnummer zu *dieser* Disk finden
  local INDEX SLOTNUM
  read INDEX SLOTNUM < <(sg_ses -p aes "$SES_DEV" 2>/dev/null \
    | awk -v want="$DISK_SAS" '
        /Element index:/             {idx=$3; gsub(":","",idx); slot=""}
        /device slot number:/        {slot=$5}
        /SAS address:/ && $3 ~ /^0x/  {sas=$3; if (tolower(sas)==tolower(want)) {print idx, slot; exit}}
      ')

  # Fallback über physische Slotnummer aus sysfs (falls vorhanden)
  if [[ -z "${INDEX:-}" || -z "${SLOTNUM:-}" ]]; then
    local p="/sys/class/block/${BASE_NOPART}/device"
    local enc=( "$p"/enclosure_device* )
    if [[ -e "${enc[0]}" && -r "${enc[0]}/slot" ]]; then
      local PHY_SLOT; PHY_SLOT="$(cat "${enc[0]}/slot" 2>/dev/null || true)"
      if [[ -n "$PHY_SLOT" ]]; then
        SLOTNUM="$PHY_SLOT"
        INDEX="$(sg_ses -p aes "$SES_DEV" 2>/dev/null \
          | awk -v slot="$PHY_SLOT" '
              /Element index:/ {idx=$3; gsub(":","",idx)}
              /device slot number:/ {if ($5==slot) {print idx; exit}}
            ')"
      fi
    fi
  fi
```

```
  # Es kann vorkommen, dass SLOTNUM leer bleibt → ist ok; dann nur Index nutzen
  [[ -n "$INDEX" ]] || { echo "SES-Index konnte nicht ermittelt werden." >&2; return 1; }
  echo "$SES_DEV" "$INDEX" "${SLOTNUM:-}" "$HOST" "$BUS" "$TARGET" "$LUN"
}

# ----------------------
# LED via SES (sg_ses)
# ----------------------
led_ses() {
  local mode="$1" SES_DEV="$2" INDEX="$3" SLOTNUM="${4:-}"
  need sg_ses
  local slot_label=""
  [[ -n "$SLOTNUM" ]] && slot_label=" (Slot $SLOTNUM)"
  case "$mode" in
    on)
      sg_ses --index="$INDEX" --set=ident "$SES_DEV"
      echo "LED an${slot_label}, SES index $INDEX via $SES_DEV – $DISK"
      ;;
    off)
      sg_ses --index="$INDEX" --clear=ident "$SES_DEV"
      echo "LED aus${slot_label}, SES index $INDEX via $SES_DEV – $DISK"
      ;;
    blink)
      sg_ses --index="$INDEX" --set=ident "$SES_DEV"
      echo "LED an${slot_label} – blinke ${BLINK_SECS}s (SES index $INDEX)…"
      sleep "$BLINK_SECS"
      sg_ses --index="$INDEX" --clear=ident "$SES_DEV"
      echo "LED aus."
      ;;
  esac
}

# ---------------------------------------------
# Methode wählen & LED einschalten (falls gew.)
# ---------------------------------------------
SES_DEV=""; INDEX=""; SLOTNUM=""; HOST=""; BUS=""; TARGET=""; LUN=""
OLDDEV=""

if (( USE_LED )); then
  case "$METHOD" in
```

```
    symlink)
      led_symlink "$([[ $BLINK_SECS -gt 0 ]] && echo blink || echo on)" \
        || { echo "Symlink-LED nicht verfügbar."; exit 1; }
      # H:B:T:L & OLDDEV für delete/scan:
      need lsscsi
      LINE="$(lsscsi -g | awk -v d="$DISK" '{if ($(NF-1)==d) {print; exit}}')"
      [[ -n "$LINE" ]] || { echo "Konnte $DISK nicht in 'lsscsi -g' finden."; exit 1; }
      OLDDEV="$(awk '{print $(NF-1)}' <<<"$LINE")"
      if [[ "$LINE" =~ \[([0-9]+):([0-9]+):([0-9]+):([0-9]+)\] ]]; then
        HOST="${BASH_REMATCH[1]}"; BUS="${BASH_REMATCH[2]}"; TARGET="${BASH_REMATCH[3]}";
LUN="${BASH_REMATCH[4]}"
      else
        echo "Konnte SCSI-Adresse nicht parsen."; exit 1
      fi
      ;;
    ses)
      read SES_DEV INDEX SLOTNUM HOST BUS TARGET LUN < <(get_ses_index) || exit 1
      led_ses "$([[ $BLINK_SECS -gt 0 ]] && echo blink || echo on)" "$SES_DEV" "$INDEX" "$SLOTNUM"
      # OLDDEV zusätzlich erfassen (nur Info)
      need lsscsi
      LINE="$(lsscsi -g | awk -v d="$DISK" '{if ($(NF-1)==d) {print; exit}}')"
      [[ -n "$LINE" ]] && OLDDEV="$(awk '{print $(NF-1)}' <<<"$LINE")" || true
      ;;
    auto)
      if ! led_symlink "$([[ $BLINK_SECS -gt 0 ]] && echo blink || echo on)"; then
        echo "(Auto) Fallback auf SES..."
        read SES_DEV INDEX SLOTNUM HOST BUS TARGET LUN < <(get_ses_index) || exit 1
        led_ses "$([[ $BLINK_SECS -gt 0 ]] && echo blink || echo on)" "$SES_DEV" "$INDEX" "$SLOTNUM"
      else
        # Symlink-Weg erfolgreich: H:B:T:L & OLDDEV besorgen
        need lsscsi
        LINE="$(lsscsi -g | awk -v d="$DISK" '{if ($(NF-1)==d) {print; exit}}')"
        [[ -n "$LINE" ]] || { echo "Konnte $DISK nicht in 'lsscsi -g' finden."; exit 1; }
        OLDDEV="$(awk '{print $(NF-1)}' <<<"$LINE")"
        if [[ "$LINE" =~ \[([0-9]+):([0-9]+):([0-9]+):([0-9]+)\] ]]; then
          HOST="${BASH_REMATCH[1]}"; BUS="${BASH_REMATCH[2]}"; TARGET="${BASH_REMATCH[3]}";
LUN="${BASH_REMATCH[4]}"
        else
          echo "Konnte SCSI-Adresse nicht parsen."; exit 1
        fi
```

```
      fi
      ;;
    *) echo "Unbekannte Methode: $METHOD"; exit 1 ;;
  esac
else
  # Keine LED: wir brauchen nur H:B:T:L & OLDDEV
  need lsscsi
  LINE="$(lsscsi -g | awk -v d="$DISK" '{if ($(NF-1)==d) {print; exit}}')"
  [[ -n "$LINE" ]] || { echo "Konnte $DISK nicht in 'lsscsi -g' finden."; exit 1; }
  OLDDEV="$(awk '{print $(NF-1)}' <<<"$LINE")"
  if [[ "$LINE" =~ \[([0-9]+):([0-9]+):([0-9]+):([0-9]+)\] ]]; then
    HOST="${BASH_REMATCH[1]}"; BUS="${BASH_REMATCH[2]}"; TARGET="${BASH_REMATCH[3]}";
LUN="${BASH_REMATCH[4]}"
  else
    echo "Konnte SCSI-Adresse nicht parsen."; exit 1
  fi
fi

# -------------
# Deregister
# -------------
echo ">>> Entferne $DISK (SCSI ${HOST}:${BUS}:${TARGET}:${LUN})"
echo 1 | sudo tee "/sys/class/scsi_device/${HOST}:${BUS}:${TARGET}:${LUN}/device/delete" >/dev/null || true

# -------------
# Prompt / Swap (ohne Slotnummer)
# -------------
if (( PROMPT )); then
  read -p ">>> Bitte neue Festplatte einsetzen und ENTER drücken... " _
fi

# -------------
# Rescan (gezielt für den Target-Slot)
# -------------
echo ">>> Rescanne Slot (Target ${TARGET}) über host${HOST}..."
echo "${HOST} ${BUS} ${TARGET}" | sudo tee "/sys/class/scsi_host/host${HOST}/scan" >/dev/null

# -------------
# Neues Device finden (by H:B:T:L)
# -------------
```

```
echo ">>> Warte auf neues Blockdevice im Slot (Target ${TARGET})..."
need lsscsi
NEWDISK=""
for i in {1..20}; do
  NEWLINE="$(lsscsi -g | awk -v h="$HOST" -v b="$BUS" -v t="$TARGET"
'$1==sprintf("[%d:%d:%d:0]",h,b,t){print; exit}')"
  if [[ -n "$NEWLINE" ]]; then
    NEWDISK="$(awk '{print $(NF-1)}' <<<"$NEWLINE")"
    echo ">>> Neues Device: $NEWDISK  ($NEWLINE)"
    break
  fi
  sleep 1
done
if [[ -n "$NEWDISK" ]]; then
  if [[ -n "$OLDDEV" ]]; then
    if [[ "$NEWDISK" == "$OLDDEV" ]]; then
      echo ">>> Hinweis: Laufwerksbuchstabe unverändert geblieben ($NEWDISK) – alles ok."
    else
      echo ">>> Hinweis: Buchstabe wechselte: alt=$OLDDEV neu=$NEWDISK (normal bei Hot-Swap)."
    fi
  fi
else
  echo "Hinweis: Im Zeitfenster kein neues Device sichtbar. Entweder Treiber/Controller langsam, oder der Slot
wurde noch nicht neu gemeldet. Prüfe lsscsi/Logs."
fi

# -------------
# LED aus
# -------------
if (( USE_LED )); then
  case "$METHOD" in
    symlink)
      led_symlink off || true           # nutzt den gecachten locate-Pfad
      ;;
    ses)
      led_ses off "$SES_DEV" "$INDEX" "$SLOTNUM" || true
      ;;
    auto)
      if [[ -n "${SES_DEV:-}" && -n "${INDEX:-}" ]]; then
        led_ses off "$SES_DEV" "$INDEX" "$SLOTNUM" || true
```

```
      else
        led_symlink off || true
      fi
      ;;
  esac
fi

echo ">>> Fertig."
```

Nun noch ausführbar machen

```
chmod +x /usr/local/bin/scsi-hotswap.sh
```

Hilfe/Benutzung:

```
Usage:
  scsi-hotswap [--method auto|symlink|ses] [--blink N|--no-led] [--no-prompt] /dev/sdX|sdX|sdXn

Optionen:
  --method M     M = auto (default) | symlink | ses
  --blink N      LED vor dem Tausch N Sekunden blinken (statt dauerhaft an)
  --no-led       keine LED-Steuerung
  --no-prompt    nicht auf ENTER warten (setze voraus, dass neue Platte schon steckt)

Beispiele:
  scsi-hotswap sdi
  scsi-hotswap --method symlink --blink 8 sdg8
  scsi-hotswap --method ses --no-prompt sdm
```

Ausgabe:

```
root@backupsrv001:~# scsi-hotswap.sh /dev/sdq
>>> Entferne /dev/sdq (SCSI 3:0:17:0)
>>> Bitte neue Festplatte in Slot 17 einsetzen...
Weiter mit ENTER, wenn neue Platte steckt.
>>> Rescanne Slot 17...
>>> Neue Device-Liste:
```

Version #11
Erstellt: 15 August 2025 14:34:06 von Admin
Zuletzt aktualisiert: 15 August 2025 17:24:12 von Admin