```
#!/usr/bin/env bash
```
Shebang: tells the OS to run this script using the bash interpreter found via env.

```
set -euo pipefail
```
Enables strict error handling: -e exit on error, -u error on undefined vars, pipefail catches pipeline failures.

```
if [[ $# -lt 1 ]]; then
```
Checks if fewer than one argument was supplied.

```
  echo "Usage: $0 <syslog_file>"
```
Prints usage instructions.

```
  exit 1
```
Exits the script because a required argument is missing.

```
fi
```
Ends the if-statement for argument checking.

```
LOG_FILE="$1"
```
Stores the first argument as the log file path.

```
if [[ ! -r "$LOG_FILE" ]]; then
```
Checks whether the target log file is readable.

```
  echo "File ’$LOG_FILE’ is not readable"
```
Prints an error if the file cannot be read.

```
  exit 1
```
Exits because the file cannot be used.

```
fi
```
Ends the readability check.

```
KEYWORDS=("ERROR" "WARNING" "FATAL" "CRITICAL")
```
Defines an array of keywords the script will search for.

```
declare -A IP_COUNTS
```
Creates an associative array that maps IP â count.

```
declare -A IP_KEYWORDS
```
Creates an associative array that maps IP â keywords found.

```
find_ip_in_line() {
```
Starts a function that extracts an IP address from a log line.

```
  local line="$1"
```
Stores the function’s input line in a local variable.

```
  local ip
```
Declares a local variable to hold the extracted IP.

```
  ip=$(grep -Eo ’([0-9]{1,3}\.){3}[0-9]{1,3}’ <<< "$line" | head -n 1 || true)
```
Uses grep with a regex to extract the first IPv4 address found in the line.

```
  if [[ -n "$ip" ]]; then
```

Checks whether an IP was successfully found.

```
    echo "$ip"
```
Outputs the extracted IP to the caller.

```
  fi
```
Ends the if-block for IP presence.

```
}
```
Ends the IP extraction function.

```
find_keywords_in_line() {
```
Starts a function that extracts matching keywords from a line.

```
  local line="$1"
```
Input line is stored locally.

```
  local found=()
```
Local array to store any matching keywords.

```
  for kw in "${KEYWORDS[@]}"; do
```
Iterates over each defined keyword.

```
    if [[ "${line^^}" == *"${kw^^}"* ]]; then
```
Performs case-insensitive keyword search.

```
      found+=("$kw")
```
Adds matched keyword to the found list.

```
    fi
```
Ends the if-statement checking keyword presence.

```
  done
```
Ends the keyword loop.

```
  if [[ ${#found[@]} -gt 0 ]]; then
```
If one or more keywords were found...

```
    echo "${found[@]}"
```
Output space-separated list of matched keywords.

```
  fi
```
Ends the if-block.

```
}
```
Ends the keyword extraction function.

```
while IFS= read -r line; do
```
Reads the log file line-by-line without trimming whitespace.

```
  ip=$(find_ip_in_line "$line" || true)
```
Extracts an IP from the current line.

```
  [[ -z "$ip" ]] && continue
```
Skips the line if no IP was detected.

```
  kws=$(find_keywords_in_line "$line" || true)
```
Extracts keywords from the line.

```
  [[ -z "$kws" ]] && continue
```
Skips the line if no relevant keywords were found.

```
  current_count=${IP_COUNTS["$ip"]:-0}
```
Gets the current count for this IP (defaulting to 0).

```
  IP_COUNTS["$ip"]=$(( current_count + 1 ))
```
Increments the count for this IP.

```
  for kw in $kws; do
```
Loops through all keywords found in this line.

```
    existing="${IP_KEYWORDS[$ip]:-}"
```
Gets the existing keyword list for this IP.

```
    if [[ " $existing " != *" $kw "* ]]; then
```
Checks if this keyword is not already stored.

```
      IP_KEYWORDS["$ip"]="$existing $kw"
```
Appends the keyword to the list for this IP.

```
    fi
```
Ends keyword duplication check.

```
  done
```
Ends keyword loop.

```
done < "$LOG_FILE"
```
Finishes reading all lines from the file.

```
DATE_PART=$(date ’+%d-%b-%Y_%H-%M’ | tr ’A-Z’ ’a-z’)
```
Generates a lowercase timestamp containing date and time.

```
REPORT_FILE="report-${DATE_PART}.rep"
```
Names the output report file.

```
{
```
Begins a group of commands whose output goes to the report file.

```
  echo "********************************************************************************"
```
Writes a decorative header line.

```
  echo "Report created at $(date +%H:%M)"
```
Writes the time the report was generated.

```
  echo
```
Outputs a blank line for spacing.

```
  for ip in "${!IP_COUNTS[@]}"; do
```
Iterates through each IP address recorded.

```
    count=${IP_COUNTS[$ip]}
```
Fetches how many lines were associated with this IP.

```
    kws=${IP_KEYWORDS[$ip]}
```
Fetches the keywords associated with this IP.

```
    kws=$(echo "$kws" | xargs)
```
Trims extra whitespace from keywords.

```
    echo "${ip} address appeared in ${count} lines."
```
Prints how often the IP appeared.

```
    echo "keywords appeared: ${kws}"
```
Prints the keywords for that IP.

```
    echo
```
Prints a blank line for readability.

```
  done
```
Ends the per-IP loop.

```
  echo "*************************************************************************************"
```
Writes the closing decorative line.

```
} > "$REPORT_FILE"
```
Redirects the entire report block into the report file.

```
echo "file name: $REPORT_FILE"
```
Prints the generated report file name for the user.

```
cat "$REPORT_FILE"
```
Displays the entire report to the terminal.