↑ 목록으로 ← 이전 단원 다음 단원 →

4단원. 쉘 스크립트로 자동화하기

> 대상: 전공자 · 정보처리기사 취득자 | 목적: 인프라 직무(네트워크·클라우드·시스템·보안) 기술 면접 대비 중요도: ★★★ 최빈출 · ★★ 자주 출제 · ★ 기본 개념

> **Q1.** 로그 로테이션(**log rotation**)이 왜 필요한지, 그리고 **logrotate**의 기본 동작 방식을 설명하세요. ★★

답안 서버가 계속 운영되면 애플리케이션과 시스템 로그가 무한정 쌓여 디스크 공간을 고갈시키고, 파일 하나가 지나치게 커지면 `grep`이나 열람 도구로 조회하는 속도도 느려집니다. 로그 로테이션은 일정 크기나 주기가 되면 현재 로그 파일을 백업용 이름으로 옮기고(예: `access.log`를 `access.log.1`로) 새 빈 파일을 생성해 애플리케이션이 계속 로그를 쓸 수 있게 하는 기법입니다. 리눅스에서는 **`logrotate`** 데몬/도구가 `/etc/logrotate.conf`와 `/etc/logrotate.d/` 아래의 설정에 따라 이 작업을 자동으로 수행하며, 압축(`compress`), 보관 개수(`rotate N`), 로테이션 주기(`daily`/`weekly`), 로테이션 후 프로세스에 신호를 보내 로그 파일 핸들을 다시 열게 하는 `postrotate` 스크립트 등을 설정할 수 있습니다. 애플리케이션이 로그 파일을 이미 열어 놓은 상태에서 파일명만 바뀌면 계속 옛 inode에 쓰게 되므로, `postrotate`에서 `SIGHUP` 등을 보내 파일을 다시 열도록 하는 처리가 중요합니다.

관련 개념 logrotate.d, postrotate, SIGHUP 재오픈, compress/rotate 옵션, 디스크 풀 방지

> **Q2.** 배치(**batch**) 작업을 설계할 때 고려해야 할 사항을 설명하세요. ★★

답안 배치 작업은 사람의 개입 없이 정해진 시각이나 조건에 자동으로 실행되므로, 먼저 멱등성(**idempotency**)을 고려해 같은 작업이 중복 실행되어도 결과가 왜곡되지 않도록 설계해야 합니다. 실행 중 다른 인스턴스가 겹쳐 실행되는 것을 막기 위해 락 파일(**lock file**)이나 `flock` 명령을 사용해 동시 실행을 방지하는 것이 일반적입니다. 실패했을 때를 대비해 로그를 남기고 알림(메일, 슬랙 등)을 보내는 처리, 그리고 재시도 로직을 넣어 일시적 오류에 강건하게 만드는 것이 중요합니다. 또한 작업 실행 시간이 다음 실행 주기를 넘기지 않도록 타임아웃을 설정하고, 처리 대상 데이터가 예상과 다를 경우(빈 파일, 형식 오류 등) 안전하게 중단할 수 있는 검증 로직도 함께 설계해야 합니다.

관련 개념 flock 락 파일, 멱등성, 타임아웃, 실패 알림, 재시도 로직

> **Q3. grep**의 주요 옵션과 활용법을 설명하세요. ★★★

답안 `grep`은 파일이나 표준입력에서 지정한 패턴과 일치하는 줄을 찾아 출력하는 텍스트 검색 도구입니다. `-i`는 대소문자를 구분하지 않고, `-v`는 패턴과 일치하지 않는 줄만 역으로 출력하며, `-r`(또는 `-R`)은 디렉토리를 재귀적으로 탐색합니다. `-n`은 일치한 줄의 번호를 함께 출력하고, `-c`는 일치한 줄의 개수만 세며, `-l`은 패턴이 포함된 파일명만 나열합니다. `-E`(또는 `egrep`)를 사용하면 확장 정규표현식을 사용할 수 있어 `+`, `?`, `|` 같은 메타문자를 이스케이프 없이 바로 쓸 수 있고, `-A/-B/-C` 옵션으로 일치한 줄의 앞뒤 컨텍스트 줄까지 함께 출력할 수 있어 로그 분석 시 매우 유용합니다.

관련 개념 -i/-v/-r/-n, 확장 정규표현식(-E), -A/-B/-C 컨텍스트, egrep

Q4. sed의 기본 동작 방식과 대표적인 치환 사용법을 설명하세요. ★★

답안 `sed` **(Stream EDitor)**는 입력을 한 줄씩 읽어 패턴 공간(pattern space)에 담아 지정된 편집 명령을 적용한 뒤 결과를 출력하는 스트림 편집 도구입니다. 가장 많이 쓰이는 치환 명령은 `sed 's/패턴/대체문자열/g'` 형태로, `s` 는 substitute(치환)를 의미하고 끝의 `g` (global) 플래그는 한 줄 안의 모든 일치 항목을 치환하며 생략하면 각 줄의 첫 번째 일치 항목만 치환합니다. 원본 파일을 직접 수정하려면 `-i` 옵션(`sed -i 's/foo/bar/g' file.txt`)을 사용하며, macOS의 BSD sed는 `-i ''` 처럼 백업 확장자를 명시해야 하는 등 GNU sed와 옵션 차이가 있어 이식성에 유의해야 합니다. 특정 줄 범위만 처리하려면 `sed -n '5,10p' file` 처럼 주소(라인 번호나 정규식)를 지정할 수 있고, 여러 설정 파일의 특정 값을 일괄 치환하는 자동화 스크립트에서 자주 사용됩니다.

관련 개념 sed -i, s/pattern/repl/g, GNU vs BSD sed 차이, 라인 주소 지정

Q5. awk의 동작 방식과 필드 처리 방법을 설명하세요. ★★★

답안 `awk` 는 입력을 줄 단위로 읽고 각 줄을 공백(기본 구분자, `-F` 로 변경 가능) 기준으로 필드(field)로 나눈 뒤, 패턴에 매칭되는 줄에 대해 지정된 액션을 실행하는 텍스트 처리 언어입니다. 필드는 `$1` , `$2` 처럼 순서대로 접근하고, `$0` 은 현재 줄 전체를 의미하며, `NR` 은 현재까지 읽은 줄 번호, `NF` 는 현재 줄의 필드 개수를 담고 있습니다. 예를 들어 `awk -F: '{print $1, $3}' /etc/passwd` 는 콜론(`:`)을 구분자로 사용자명(1번 필드)과 UID(3번 필드)를 출력합니다. awk는 `BEGIN{}` 과 `END{}` 블록을 지원해 처리 전/후에 초기화나 합계 출력 같은 작업을 할 수 있고, 변수 저장과 조건문·반복문까지 지원하는 하나의 미니 프로그래밍 언어이기 때문에 `grep` 이나 `sed` 보다 복잡한 로그 집계(예: 특정 필드 합산, 카운트)에 훨씬 강력합니다.

관련 개념 -F 구분자, NR/NF, BEGIN/END 블록, $0/$1 필드 참조, 로그 집계

> **Q6.** 정규표현식의 기본 메타문자를 설명하세요. ★★

답안 정규표현식에서 `.`은 임의의 문자 하나를, `*`는 바로 앞 문자가 0회 이상 반복됨을, `+`(확장 정규식)는 1회 이상 반복을, `?`는 0회 또는 1회를 의미합니다. `^`는 줄의 시작, `$`는 줄의 끝을 나타내는 앵커(anchor)이며, `[]`는 문자 집합(예: `[0-9]`는 숫자 하나)을, `[^...]`는 부정 집합을 의미합니다. **`{n,m}`**은 반복 횟수 범위를 지정하고, `|`(확장 정규식)는 OR 조건을 나타내 여러 패턴 중 하나라도 일치하면 매칭됩니다. `grep`, `sed`, `awk` 모두 정규표현식을 사용하지만 기본 정규식(BRE)과 확장 정규식(ERE)에서 `+`, `?`, `|`, `{}` 등을 이스케이프해야 하는지가 도구별로 다르므로, `grep -E` 처럼 확장 모드 옵션 유무를 확인하는 것이 중요합니다.

관련 개념 BRE vs ERE, 앵커(^/$), 문자 집합 [], 반복 {n,m}, grep -E

> **Q7.** 쉘 스크립트에서 에러 처리 기법(**set -e, trap, set -u**)을 설명하세요. ★★★

답안 기본적으로 bash 스크립트는 어떤 명령이 실패(0이 아닌 종료 코드)해도 다음 줄을 계속 실행하는데, **`set -e`**를 스크립트 상단에 선언하면 명령이 실패하는 즉시 스크립트 실행을 중단시켜 오류를 조기에 감지할 수 있습니다. **`set -u`**는 선언되지 않은 변수를 참조할 때 오류로 처리해 오타로 인한 빈 변수 사용을 방지하고, 이 둘과 앞서 다룬 `set -o pipefail`을 함께 사용하는 `set -euo pipefail`은 자동화 스크립트에서 사실상 표준적인 안전장치로 권장됩니다. **`trap`**은 스크립트가 특정 시그널을 받거나 종료(`EXIT`)될 때 실행할 명령을 등록하는 기능으로, 예를 들어 `trap 'rm -f "$TMPFILE"' EXIT`를 걸어두면 스크립트가 정상 종료되든 중간에 실패하든 항상 임시 파일을 정리하도록 보장할 수 있습니다. 이런 기법들은 배치 작업이 어중간한 상태로 실패해 데이터 정합성을 해치는 상황을 방지하는 데 핵심적입니다.

관련 개념 set -euo pipefail, trap EXIT, 임시 파일 정리, 선언되지 않은 변수 오류

> **Q8. cron**과 쉘 스크립트를 연계할 때 주의할 점을 설명하세요. ★★

답안 cron으로 실행되는 스크립트는 사용자가 로그인해서 얻는 셸 환경과 달리 매우 최소한의 환경변수(PATH 등)만 가진 채로 실행되므로, 스크립트 안에서 사용하는 명령어나 파일 경로는 반드시 절대경로로 작성하거나 스크립트 초반에 `PATH`를 명시적으로 설정해야 합니다. cron은 기본적으로 표준출력/에러를 등록된 관리자 메일로 보내려 시도하는데, 메일 시스템이 구성되어 있지 않으면 출력이 유실될 수 있으므로 `>> /var/log/myjob.log 2>&1`처럼 로그 파일로 명시적으로 리다이렉션하는 것이 안전합니다. 또한 스크립트 자체에서 앞서 다룬 종료 코드와 로깅을 남겨야 실패를 사후에 파악할 수 있고, 실행 시간이 예상보다 길어져 다음 스케줄과 겹치지 않도록 `flock`으로 중복 실행을 방지하는 것이 좋습니다. 시간대(timezone) 설정이 서버와 다르면 의도치 않은 시각에 실행될 수 있다는 점도 확인이 필요합니다.

관련 개념 PATH 명시, 리다이렉션으로 로그 남기기, flock 중복 방지, 타임존 확인

> **Q9. find**와 **xargs**를 조합해 자동화 작업을 수행하는 방법을 설명하세요. ★

답안 **`find`**로 조건에 맞는 파일 목록을 찾은 뒤, **`xargs`**를 이용해 그 목록을 다른 명령의 인자로 넘겨 일괄 처리하는 패턴은 대량 파일 작업 자동화에서 매우 흔합니다. 예를 들어
`find /var/log -name "*.log" -mtime +30 -print0 | xargs -0 rm -f`는 30일 이상 지난 로그 파일을 찾아 한꺼번에 삭제합니다. 여기서 `-print0`와 `xargs -0`을 함께 쓰는 이유는 파일명에 공백이나 개행이 포함되어 있어도 널(null) 문자를 구분자로 사용해 안전하게 분리하기 위함입니다. `find`의 `-exec` 옵션으로도 동일한 작업을 할 수 있지만(`find ... -exec rm -f {} \;`), 대상 파일이 매우 많을 경우 `xargs`가 여러 파일을 묶어 명령을 적은 횟수로 실행하기 때문에 성능상 더 유리한 경우가 많습니다.

관련 개념 -print0 / xargs -0, find -exec 비교, -mtime 조건, 성능(호출 횟수)

> **Q10.** 쉘 스크립트를 디버깅하는 방법(**set -x, shellcheck**)을 설명하세요. ★

답안 `set -x`를 스크립트에 추가하면 실행되는 각 명령어와 확장된 변수 값을 실행 전에 그대로 화면에 출력해 주어, 스크립트가 실제로 어떤 명령으로 치환되어 실행되는지 한 줄씩 추적할 수 있습니다. 특정 구간만 디버깅하고 싶다면 그 구간 앞뒤에 `set -x`와 `set +x`(디버그 종료)를 배치하면 됩니다. 실행 전에 잠재적 버그를 잡고 싶다면 정적 분석 도구인 `shellcheck`를 사용하는 것이 좋은데, 이는 인용 부호 누락, 잘못된 비교 연산자 사용, 사용하지 않는 변수 등 흔한 셸 스크립팅 실수를 규칙 기반으로 지적해 줍니다. 이 외에도 `bash -n script.sh`로 실제 실행 없이 문법 오류만 검사(syntax check)할 수 있어, CI 파이프라인에 셸 스크립트 검증 단계로 넣기에 적합합니다.

관련 개념 set -x/+x, shellcheck 정적 분석, bash -n 문법 검사, CI 통합
