🛠️ Examen 107 - Tareas Administrativas

107.2 Automatizar tareas de administración del sistema planificando trabajos

Teoría

La planificación de trabajos (job scheduling) es esencial para realizar tareas de mantenimiento, copias de seguridad, rotación de logs, actualizaciones del sistema y otras operaciones repetitivas sin intervención manual. Las utilidades principales para esto en Linux son `cron` para trabajos recurrentes y `at` para trabajos únicos.

Trabajos Recurrentes con `cron`:

`cron` es un demonio (servicio que se ejecuta en segundo plano) que lee archivos de configuración llamados "crontabs" (cron tables) y ejecuta los comandos especificados en ellos a las horas y fechas predefinidas.

1. **Archivos Crontab:**
    - **Crontabs de Usuario:** Cada usuario puede tener su propio archivo crontab para planificar trabajos personales. Estos archivos no se editan directamente, sino a través del comando `crontab`. Se almacenan típicamente en `/var/spool/cron/crontabs/` con el nombre de usuario (ej: `/var/spool/cron/crontabs/tu_usuario`). Solo root puede ver o modificar los crontabs de otros usuarios.
        - `crontab -e`: Edita el archivo crontab del usuario actual. Abre un editor de texto (generalmente `vi` o el definido en `$EDITOR`). Al guardar y salir, la sintaxis se verifica y el crontab se instala.
        - `crontab -l`: Muestra el archivo crontab del usuario actual.
        - `crontab -r`: Elimina el archivo crontab del usuario actual.
    - **Crontab del Sistema:** Un archivo crontab global para trabajos a nivel de sistema. Requiere permisos de root para editarse.
        - **Rama Debian/Ubuntu:** `/etc/crontab`. Este archivo a menudo tiene una sintaxis ligeramente extendida para incluir el usuario que ejecuta el comando.
        - **Rama Red Hat/CentOS/Fedora:** `/etc/crontab`. Similar a Debian, también puede tener el campo de usuario.
    - **Directorios de `cron` del Sistema:** Ubicaciones estándar donde los paquetes de software o el administrador del sistema pueden colocar scripts para ser ejecutados a intervalos predefinidos por el `cron` del sistema. Estos directorios son procesados por un script especial (a menudo `run-parts` o equivalente) invocado desde `/etc/crontab`.
        - **Rama Debian/Ubuntu:** `/etc/cron.hourly/`, `/etc/cron.daily/`, `/etc/cron.weekly/`, `/etc/cron.monthly/`. Cualquier script

ejecutable colocado en estos directorios será ejecutado por el `cron` del sistema a la hora/fecha correspondiente.
    - **Rama Red Hat/CentOS/Fedora:** También usan `/etc/cron.hourly/`, `/etc/cron.daily/`, `/etc/cron.weekly/`, `/etc/cron.monthly/`. La lógica es la misma. Pueden existir directorios adicionales como `/etc/cron.d/` para colocar archivos crontab completos (con sintaxis estándar de 6 campos + usuario) específicos de una aplicación. `run-parts` también se usa comúnmente aquí.
2. **Sintaxis de Crontab (para crontabs de usuario y archivos en `/etc/cron.d/`):** Cada línea de una tarea cron tiene 6 campos:`minuto hora dia_del_mes mes dia_de_la_semana comando`
    - **minuto:** (0-59)
    - **hora:** (0-23)
    - **dia_del_mes:** (1-31)
    - **mes:** (1-12 o nombre del mes)
    - **dia_de_la_semana:** (0-7, donde 0 y 7 son domingo)
    - **comando:** La línea de comando o script a ejecutar.
    - **Caracteres especiales:**
        - `*`: Cualquier valor (ej: cada minuto, cada hora).
        - `,`: Lista de valores (ej: `0,15,30,45` para cada cuarto de hora).
        - `-`: Rango de valores (ej: `9-17` para horas de 9 AM a 5 PM).
        - `/`: Incremento (ej: `*/15` para cada 15 minutos, `0-23/2` para horas pares).
        - `?`: Usado en algunos sistemas en el campo `dia_del_mes` o `dia_de_la_semana` para indicar "no especificar" (evitar conflictos entre los dos campos). No es estándar en todas partes.
        - `@reboot`: Ejecutar una vez al arrancar.
        - `@hourly`, `@daily`, `@weekly`, `@monthly`, `@yearly`: Atajos comunes.
3. **Sintaxis de Crontab del Sistema (`/etc/crontab`):** La sintaxis es similar, pero a menudo añade un campo adicional después del campo `dia_de_la_semana` para especificar el **usuario** que ejecutará el comando.`minuto hora dia_del_mes mes dia_de_la_semana usuario comando`

Trabajos Únicos con `at`:

`at` se utiliza para planificar la ejecución de comandos o scripts una sola vez en un momento especificado en el futuro. `batch` es similar a `at`, pero ejecuta el trabajo cuando el nivel de carga del sistema lo permite.

1. **`at`:**

- `at <tiempo>`: Lee comandos desde stdin (o un archivo con `-f`) hasta que pulses `Ctrl+D`, y los planifica para ejecutarse al `<tiempo>` especificado. La especificación del tiempo es muy flexible (ej: `now + 10 minutes`, `midnight tomorrow`, `2 PM Friday`, `14:00 2025-12-31`).
- Los trabajos planificados se guardan típicamente en `/var/spool/at/`.

2. **`atq`:**
   - Lista los trabajos planificados por `at` para el usuario actual. Muestra el número del trabajo, la fecha y la cola.
   - `sudo atq`: Muestra todos los trabajos de todos los usuarios.
3. **`atrm`:**
   - Elimina trabajos planificados por `at`.
   - `atrm <numero_trabajo>`: Elimina el trabajo especificado por su número (obtenido con `atq`).
4. **`batch`:**
   - Similar a `at now`, pero el trabajo se ejecuta solo cuando el promedio de carga del sistema cae por debajo de un umbral (típicamente 1.5). Útil para ejecutar tareas que consumen recursos sin impactar a los usuarios interactivos.
   - `batch`: Lee comandos desde stdin hasta `Ctrl+D` y los planifica.
   - Los trabajos de `batch` aparecen en la cola `b` en `atq`.

Control de Acceso (`cron.allow`, `cron.deny`, `at.allow`, `at.deny`):

Puedes controlar qué usuarios pueden usar `cron` y `at` mediante archivos de control de acceso en `/etc/`:

- `/etc/cron.allow`: Si existe, solo los usuarios listados en este archivo (uno por línea) pueden usar `crontab`. `root` siempre puede.
- `/etc/cron.deny`: Si `cron.allow` no existe, los usuarios listados en este archivo *no* pueden usar `crontab`. Si ninguno de los archivos existe, por defecto la mayoría de los sistemas permiten que todos los usuarios usen `crontab` (excepto aquellos listados en `/etc/passwd` que no deberían tener acceso de login).
- La lógica para `/etc/at.allow` y `/etc/at.deny` es idéntica para el comando `at`.

Consideraciones sobre Entorno:

Los trabajos de `cron` y `at` se ejecutan en un entorno de shell minimalista. Es posible que variables de entorno a las que estás acostumbrado (como un `PATH` personalizado en tu `.bashrc`) no estén disponibles. Por ello:

- Siempre usa rutas *absolutas* para comandos y archivos en scripts cron/at.
- O establece la variable `PATH` dentro del script o al principio del archivo crontab.

- Redirecciona la salida estándar y de error de tus comandos a archivos o a `/dev/null`, ya que la salida no interactiva de cron/at suele enviarse por correo electrónico al usuario propietario del trabajo (o a root). `comando > /ruta/a/log 2>&1`