|  |
| --- |
| * Ch\_1 \_ RAID 원상 복구 * Ch\_2 \_ RAID6, 10 (1+0) 구축 * Ch\_3 \_ RAID6, 10 문제 발생 테스트 * Ch\_4 \_ LVM 개념과 실습 * Ch\_5 \_ 쿼터 개념과 실습 * Ch\_6 \_ 셸의 기본과 작성법 * RAID 정상 복구 * 고장난 디스크 4개를 새 디스크로 교체 * edit virtual machien setting > 하드디스크 4개 추가      * RAID1 같은 경우는 ‘결함 허용’ 이기 때문에 잘 실행되는 디스크에 추가만 해줌 * --add 사용 (md1)      * 재 부팅 후 이름이 바뀌어 있는 상황 * 종종 있는 현상으로, /etc/fstab에 변경됨 이름으로 다시 저장하면 해결됨      * fstab 변경 후 재부팅 시 정상 작동 확인 * RAID 6와 RAID 1+ 0 개념 * RAID5보다 신뢰도를 높인 RAID6 * 신뢰도와 속도 두마리의 토끼를 잡기 위한 RAID1 +0 * RAID6은 패리티를 2개 사용하기 때문에 최소 4개의 디스크가 필요 * RAID1+0은 최소 4개 디스크가 필요 * RAID6의 공간효율은 n-2 * RAID1+0의 공간효율은 50%      * 디스크9장착 시점으로 스냅샵 이후 RAID 적용 준비 완료      * 앞선 내용과 동일하게 RAID 생성 후 마운트, fstab설정, mdadm.conf 설정 완료 * md6은 RAID6으로 기존 RAID 생성과 동일 * md10은 RAID 0+ 1 으로 RAID1을 2개 생성 후 RAID1들을 묶어 RAID0 생성 * mount 이후 재접속 시 자동 mount를 위해 fstab에 등록, mdadm.conf 등록(nameserver는 삭제)      * update-initramfs -u 명령어로 업데이트 후 재부팅      * RAID가 잘 적용되어 있는 모습 * RAID6, 10 문제 발생 테스트 * 각 2개씩 하드디스크를 고장 낸다. * 고장 후에도 데이터의 이상 여부를 확인한다.        * 4개 하드디스크 삭제 * RIAD6 (SCSI0:2, SCSI0:4) RAID10 (SCSI0:6, SCSI0:8) 삭제      * 삭제 후 실행하니 마운트 진행 X (데이터가 사라진 것은 아님)      * 수동 마운트 진행        * 디스크가 고장(삭제)된 상태인데도 불구하고 데이터는 보존되어있음 * 실제 상황에서는 꼭 데이터를 백업받아야 함 (컴퓨터가 비정상 상태이기 때문)      * /dev/md6 는 /sdb1 , /sdc1 으로만으로 잘 작동되는 것을 확인          * 각각 md2, md3, md10 상태 확인 * md2, 3는 각각 1개씩만 작동, md10은 이상 없음 확인 가능      * 정상적인 부팅이 될 수 있도록 RAID장치 해제 * mdadm --stop을 차례대로 진행하여야 함 (바깥부터/ md10 -> md2,md3) * 이후 이전 학습처럼 원상 복구 절차 진행 * LVM 개념 (1) * LVM (Logical Volume Manage) 개념 이해 * LVM 주요 기능   여러 개의 하드디스크를 합쳐서 한 개의 파일시스템으로 사용하는 것으로 필요에 따라서 다시 나눌 수 있다.  예로 2TB 용량의 하드디스크 2개를 합친 후에 다시 1TB와 3TB로 나눠서 사용할 수 있다.   * 용어   Physical Volume(물리 볼륨): /dev/sda1, /dev/sab1 등의 파티션  Volume Group(볼륨 그룹): 물리 볼륨을 합쳐서 1개의 물리 그룹으로 만드는 것  Logical Volume(논리 볼륨): 볼륨 그룹을 1개 이상으로 나눠서 논리 그룹으로 나눈 것   * LVM을 구현하려고 하드디스크 2개를 추가한 구성도      * LVM 구현 실습 흐름도        * 하드디스크 2개 추가 (3GB, 2GB 총 5GB)      * sdb, sdc fdisk 명령어 작업 * type code ‘8e’ 입력하여 LVM 으로 생성      * 물리 볼륨 명령어인 pvcreate 사용 위해 lvm2 install 진행      * sdb1 , sdc1 을 물리 볼륨으로 변경      * 두개의 물리 볼륨을 묶어서 하나의 볼륨 그룹으로 생성 * vgcreate myVG /dev/sdb1 /dev/sdc1명령어 사용하여 볼륨 그룹 생성 * vgdisplay 명령으로 현재 볼륨 그룹들을 확인 가능      * 불륨 그룹을 나눠서 논리 그룹 생성 * lvcreate 명령어로 기존 볼륨 그룹을 나눠서 생성   lvcerate --size 1G --name myLG1 myVG | 크기는 ‘ ’ name 은 ‘ ‘ [ ] 에서 나눈다   * 마지막 그룹을 생성할 때 나머지 공간을 전부 활용   lvcreate --extends 100%FREE --name myLG3 myVG     * 생산이 성공적으로 완성      * 디렉터리 생성 후 마운트 진행 * mkfs.ext4 명령어로 포맷진행 * mkdir 명령어로 디렉터리 3개 생성 (lvm1, 2, 3) * mount 명령어로 마운트 진행      * fstab 수정하여 자동 마운트 설정      * 재부팅 후 정상 작동 확인 * 사용자별 공간 할당 -쿼터 * 쿼터(Quota) 개념 * 파일시스템마다 사용자나 그룹이 생성할 수 있는 파일의 용량 및 개수를 제한하는 것 * 파일시스템을 “/”로 지정하는 것보다는, 별도의 파일시스템을 지정해서 해당 부분을 쓰도록 하는 것이 좋음 * “/”파일시스템을 많은 사용자가 동시에 사용하게 되면, 우분투 서버를 운영하기 위해서 디스크를 읽고 쓰는 작업과 일반 사용자가 디스크를 읽고 쓰는 작업이 동시에 발생하므로 전반적으로 시스템의 성능이 저하됨 * 사용자를 만들고 해당 사용자에게 공간 할당        * 기존과 동일하게 진행하여 기본 리눅스 타입으로 생성 * 포맷진행 > 디렉터리생성 > 마운트 > fstab 등록        * adduser 명령어로 사용자 2개 추가 (john , daniel // passwd는 동일하게 ‘1234’) * adduser --home /userhome/john john      * 생성한 일반 디스크를 쿼터용으로 변경 * defaults,usrjqota=auqota=aquta.user, jpfmt=vfsv0 추가      * remount로 재부팅 효과 * 재부팅을 해도 좋지만 remount로 재부팅의 효과를 줄 수 있음 (다시 마운트함) * mount --options remount [디렉터리명]      * 쿼터형으로 마운트가 성공적으로 적용된 것을 확인      * 쿼터 패키지 install 진행      * 쿼터 작동 성공      * 사용자 공간 제어 * edquota -u john 명령어로 john 사용자 설정 * blocks, soft, hard 로 공간을 제어할 수 있다. (단위는 kb) // 0은 무제한으로 사용 가능 * soft는 초과 할 수는 있지만 경고 / hard는 절대로 초과할 수 없다      * 사용자 공간 제한 테스트를 위해 john으로 접속      * cp명령어가 진행되다가 공간 제한에 걸리자 실행 취소가 된다 * quota 명령어로 사용자는 자신의 할당량을 확인 가능 (quota = soft / limit = hard)      * root 사용자는 repqouta [디렉토리명]명령어로 사용자 별 사용량 확인 가능 * 셸의 기본과 작성법 * 우분투의 bash 셸 (터미널) * 기본 셸은bash(Bourne Again Shell:’배시 셸’) * bash 셸의 특징 * Alias 기능(명령어 단축 기능) * History 기능(위/아래 화살표키) * 연산 기능 * job Control 기능 * 자동 이름 완성 기능 (탭키) * 프롬프트 제어 기능 * 명령 편집 기능 * 셸의 명령문 처리 방법 * (프롬프트) 명령어 [옵션…][인자…] * 예) # rm -fg /mydir * 환경 변수 * “echo $환경변수이름” 으로 확인 가능 * “export 환경변수=값” 으로 환경 변수의 값을 변경 * 주요 환경변수        * HOME 과 PATH 사용 확인 * 셸 스크립트 프로그래밍 * c언어와 유사하게 프로그래밍이 가능 * 변수, 반복문, 제어문 등의 사용이 가능 * 별도로 컴파일하지 않고 텍스트 파일 형태로 바로 실행 * vi나 gedit으로 작성이 가능 * 리눅스의 많은 부분이 셸 스크립트로 작성되어 있음      * 셸 스크립트의 작성과 실행      * 실행방법  1. “sh <스크립트파일>”로 실행 2. “chmod +x <스크립트 파일>” 명령으로 실행 가능 속성으로 변경한 후에, “./<스크립트파일>” 명령으로 실행        * gedit로 name.sh을 생성 뒤 작성 후 실행 * 사용자이름과 홈 디렉터리가 나타나는 기능 코딩 * sh <스크립트파일> 로 실행 (sh name.sh) |