컴퓨터학부 20203123 이중곤 비밀번호: 010126

- 0. 리눅스 커널 및 리눅스 배포판의 이해

- 리눅스 배포판, 리눅스 커널, Ubuntu LTS 버전, Ubuntu 정규버전 용어를 설명하시오.

* 리눅스 커널
  + 하드웨어와 소프트웨어 간의 상호작용을 관리하는 운영체제의 핵심 부분
  + 메모리, 프로세스, 드라이버, 파일 시스템 등을 관리함
* 리눅스 배포판
  + 리눅스 커널을 기반으로 패키지 매니저, 개발 툴, 유틸리티 프로그램, 데스크톱 환경 등의 소프트웨어를 포함하여 배포되는 운영체제
  + Ubuntu, Fedora, Debian, Cent OS 등이 있음
* Ubuntu LTS 버전
  + 리눅스 배포판 중 하나인 Ubuntu의 장기 지원(Long-Term Support)이 제공되는 버전
  + 일반적으로 2년마다 출시되며, 각 LTS 버전은 5년간 보안 업데이트와 패치 지원을 받음
* Ubuntu 정규 버전
  + LTS 버전과 달리 매년 4월과 10월에 출시되는 버전
  + LTS 버전에 비해 최신 기능과 개선된 요소들이 빠르게 반영됨

- 1. /proc 파일 시스템의 이해

- (1) more /proc/cpuinfo 명령을 실행하고 processor, cores 용어를 설명하시오.

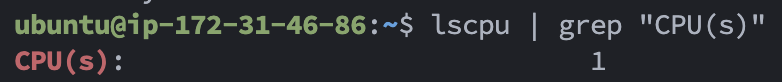
텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

* processor
  + 컴퓨터 프로그램의 명령어를 해석하고 실행하는 CPU를 지칭
  + 현대의 프로세서는 여러 개의 코어로 구성됨
* cores
  + 프로세서 내부의 독립적인 처리 유닛
  + 각 코어는 독립적으로 명령어를 실행

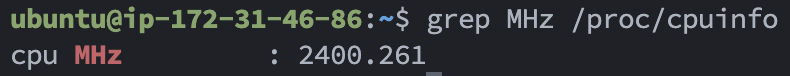
(2) (학생의) 컴퓨터 시스템에 있는 프로세스서의 개수는 몇 개인지 확인하시오.

* lscpu | grep "CPU(s)"
  + lscpu 명령어로 CPU의 하드웨어 정보를 출력한 후 시스템의 논리적 프로세스의 개수를 나타내는 “CPU(s)”를 grep 명령어로 가져와 출력
* 시스템에 있는 논리적 프로세스의 개수: 1개



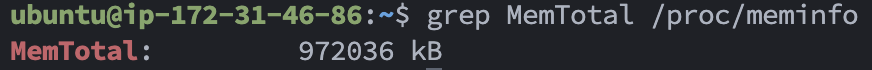
(3) 각 프로세서의 frequency는 얼마인지 확인하시오.

* grep MHz /proc/cpuinfo
  + CPU 관련 정보를 담고 있는 파일인 /proc/cpuinfo에서 CPU 동작 주파수를 나타내는 MHz를 grep 명령어로 가져와 출력
* 프로세서의 frequency: 2400MHz



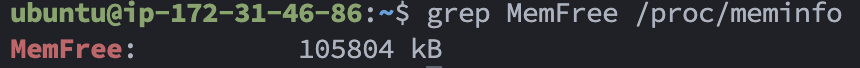
(4) (학생의) 컴퓨터 시스템의 물리적 메모리 크기는 얼마인지 확인하시오.

* grep MemTotal /proc/meminfo
  + 시스템의 메모리 상태를 보여주는 파일인 /proc/meminfo에서 시스템의 총 물리적 메모리 용량을 나타내는 MemTotal을 grep 명령어로 가져와 출력
* 물리적 메모리 크기: 972 MB



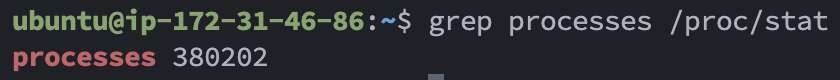
(5) 위 물리적 메모리 중 free한 크기는 얼마인지 확인하시오.

* grep MemFree /proc/meminfo
  + 시스템의 메모리 상태를 보여주는 파일인 /proc/meminfo에서 사용하지 않고 남아있는 메모리 용량을 나타내는 MemFree를 grep 명령어로 가져와 출력
* 물리적 메모리 크기: 104 MB



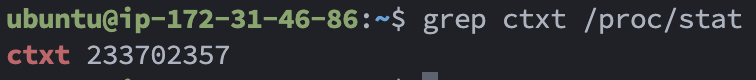
(6) 시스템 부팅 후 fork()된 프로세스의 개수는 몇 개인지 확인하시오.

* grep processes /proc/stat
  + 시스템 통계 정보를 보여주는 파일인 /proc/stat에서 시스템이 부팅된 이후 생성된 총 프로세스 수를 나타내는 processes를 grep 명령어로 가져와 출력
* 시스템이 부팅된 이후 생성된 모든 프로세스의 개수: 380,202개



(7) 부팅 후 문맥 교환(context switch)가 이루진 횟수는 얼마인지 확인하시오.

* grep ctxt /proc/stat
  + 시스템 통계 정보를 보여주는 파일인 /proc/stat에서 시스템에서 발생한 컨텍스트 스위치 총 횟수를 나타내는 ctxt를 grep 명령어로 가져와 출력
* 시스템 부팅 이후의 총 context switch 횟수: 23,370,237번



- 2. 실행 중인 프로세스의 상태 모니터링

(1) 주어진 cpu.c를 컴파일하고 아래 명령어를 실행하고 top 명령어 실행하시오.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

(2) cpu 명령을 실행하는 프로세스의 PID는 무엇인가 설명하시오.

* 무한 루프를 돌면서 CPU 자원을 가장 많이 소모하는 380682가 cpu 프로세스의 PID이다.

(3) 위 프로세스는 얼마나 많은 CPU와 메모리를 소비하는지 확인하시오.

* cpu 프로세스(380682)는 99.9%의 CPU와 0.1%의 메모리를 소비한다.

(4) 위 프로세스의 현재 상태는 무엇인지 확인하시오.

* 프로세스의 상태를 나타내는 S열의 값이 R이므로 실행 중(Running) 상태이다.

- 3. 새로운 자식 프로세스를 생성하여 사용자 명령을 실행하는 방법 이해

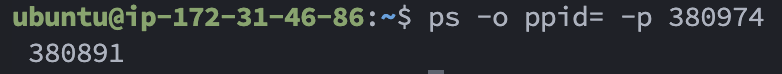
(1) cpu-print.c를 컴파일하고 쉘에서 아래 명령 차례로 실행하시오.

* ps aux | grep './cpu-print'
  + ps 명령어로 현재 실행 중인 프로세스 목록을 출력한 후 grep 명령어로 cpu-print 프로세스를 가져와 출력

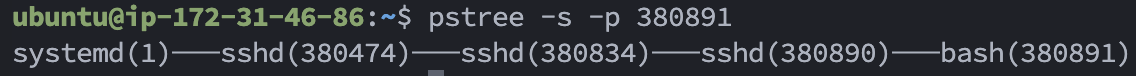


(2) cpu-print 프로세스의 부모, 즉 쉘 프로세스의 PID는 무엇인지 확인하시오? 이 pid로부터 5세대 이상 의 조상 프로세스 (또는 init 프로세스)에 도달 할 때까지 거슬러 모든 조상의 PID는 무엇인지 확인하시오.

* ps -o ppid= -p 380974
  + -p 옵션으로 cpu-print 프로세스의 PID(380974)를 지정하고 -o 옵션으로 부모 프로세스의 PID를 나타내는 ppid 필드를 지정하여 출력
* cpu-print 프로세스(380974)의 부모 PID 확인: 380891



* pstree -s -p 380891
  + 실행 중인 프로세스들을 트리 구조로 보여주는 명령어인 pstree 실행 후 -s 옵션으로 지정한 프로세스의 부모 프로세스를 보여주고 -p 옵션으로 각 프로세스의 PID도 함께 출력



(3) $ ./cpu-print> /tmp/tmp.txt & 을 실행하고 새로 생성 된 프로세스의 proc 파일 시스템 정보를 확인하시오. file descriptor 0, 1 및 2 (표준 입력, 출력 및 에러)가 가리키는 위치에 주의.

* ls -l /proc/381040/fd
  + 새로 생성 된 프로세스(381040)가 열고 있는 파일 디스크립터 목록을 출력

텍스트, 스크린샷, 폰트이(가) 표시된 사진

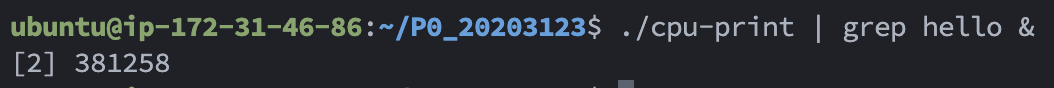
자동 생성된 설명

(4) 위 이러한 내용을 토대로, 셸에서 I/O 리디렉션을 구현하는 방법을 간단하게 설명하시오.

* > - 프로세스의 출력 스트림을 특정 파일로 변경
* e.g., ls > ls.txt
  + ls의 출력 스트림이 ls.txt 파일로 변경되므로 ls의 출력 결과가 ls.txt 파일에 저장됨
* < - 프로세스의 입력 스트림을 특정 파일로 변경
* e.g., cat < cat.txt
  + cat의 입력 스트림이 cat.txt 파일로 변경되므로 cat.txt 파일의 내용이 화면에 출력됨

(5) $ ./cpu-print | grep hello & 을 실행하고 새로 생성된 프로세스는 무엇인지 확인하시오.

* cpu-print 프로세스의 표준 출력이 grep hello 명령어의 표준 입력으로 전달된다.
* grep hello 명령어는 cpu-print가 표준 출력에 출력한 데이터에서 “hello”라는 문자열이 포함된 행을 필터링하여 출력
* & 연산자로 인해 프로세스는 백그라운드에서 실행되고 해당 프로세스의 PID는 381258



- 4. 프로세스의 가상 메모리와 실제 메모리 비교

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명(1) 주어진 두 프로그램 memory1.c 및 memory2.c를 컴파일하고 실행하시오.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

(2) 각 프로세스의 가상메모리 크기와 실메모리 크기를 캡쳐해서 비교하고, 주어진 프로그램 소스 코드와 ps 명령어를 통해 확인한 내용을 자세하게 설명하시오.

* ps -p 50019,50097 -o pid,comm,vsz,rss
  + memory1(50019), memory2(50097) 프로세스의 프로세스 ID(PID), 명령어 이름(COMM), 가상 메모리 사용량(VSZ), 실제 메모리 사용량(RSS) 정보를 표시

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

두 프로그램 모두 동일한 크기의 배열을 할당했기 때문에 VSZ(가상 메모리 크기)는 비슷하지만 memory1은 실제로 배열을 사용하지 않고 memory2는 실제로 배열에 접근하여 사용하기 때문에 RSS(실제 메모리)는 memory2 프로세스가 큰 것을 확인할 수 있다.

- 5. 프로세스가 오픈한 파일 그리고 디스크  
(1) 주어진 disk.c 및 disk1.c 프로그램을 컴파일하고 실행하시오.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

(2) disk.c 및 disk1.c 의 실행 파일 프로그램이 실행되는 동안 디스크 사용률을 측정하고, 주어진 프로그램 소스 코드와 iostat 명령어를 통해 확인한 내용을 자세하게 비교 설명하시오.

* disk 프로세스 실행 후 iostat -d 1 -p xvda 실행 결과
  + 1초 간격으로 xvda 디스크 파티션에 대한 I/O 통계를 모니터링

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

* disk1 프로세스 실행 후 iostat -d 1 -p xvda 실행 결과
  + 1초 간격으로 xvda 디스크 파티션에 대한 I/O 통계를 모니터링

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

`disk` 프로세스는 매번 다른 파일을 디스크에서 읽기 때문에 디스크로부터 읽은 데이터의 총량을 나타내는 `kB\_read`의 값이 시간이 지나도 일정하게 유지되는 것을 알 수 있다.

반면, `disk1` 프로세스는 항상 같은 파일을 반복해서 읽기 때문에 해당 파일이 메모리에 캐싱되어 일정 시간이 지나면 `kB\_read`의 값이 0으로 유지된다