상명대학교 컴퓨터과학부 "EA0014: 유닉스프로그래밍" 실습 지침서

- 실습 번호 : lab-04

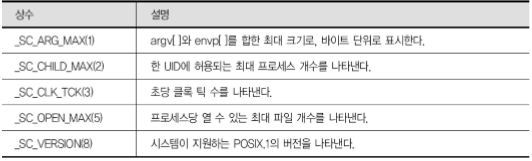
- 실습 디렉토리 : ~/unix/lab-04 이름 : 임현

- 실습 날짜 : 2017년 04월 13일 분반 : 1분반

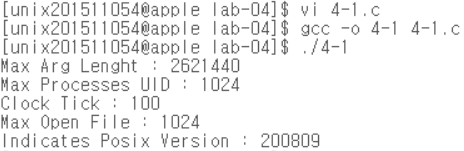
- 실습 제목 : System and Process Infos

- 실습 내용 : 학번 : 201511054

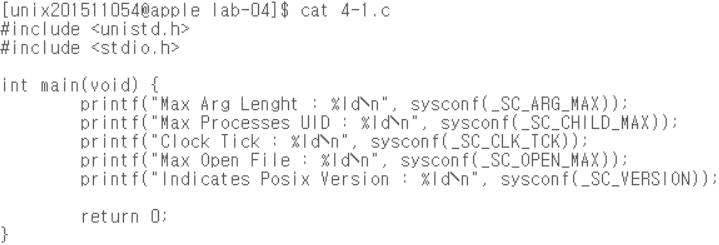
1. (4-1.c, 4-1) 교재의 예제 4-4을 수정하여 아래 표의 각 상수값을 출력하는 프로그램 4-1.c를 작성하고, 컴파일하여 4-1을 만들고, 이를 수행한 결과를 적으시오.



수행한 결과는 아래와 같고,

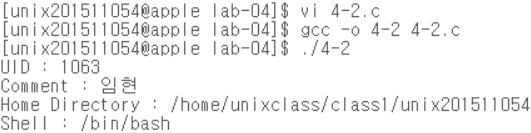


코드는 아래와 같습니다.

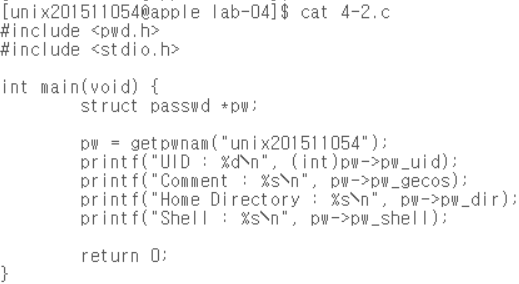


2.(4-2.c, 4-2) 교재의 예제 4-8을 수정하여 자신의 로그인 네임에 해당하는 /etc/passwd 의 엔트리를 읽어들여 pw\_uid, \*pw\_comment, \*pw\_dir, \*pw\_shell을 출력하는 프로그램 4-2.c를 작성하고, 컴파일하여 4-2를 만들고, 이를 수행한 결과를 적으시오.

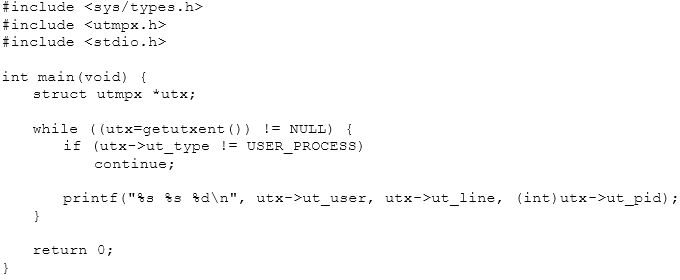
수행한 결과는 아래와 같고,



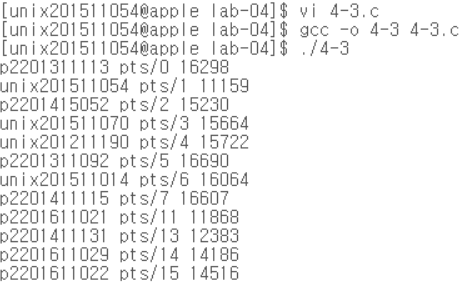
코드는 아래와 같습니다.



3.(4-3.c, 4-3) 아래와 같은 프로그램 4-3.c를 만들고 컴파일하여 수행하시오. 그리고 이 프로그램이 무엇을 하는 프로그램인지 설명하시오.



수행 결과는 아래와 같고,



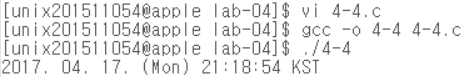
이 프로그램은 getutext 함수를 이용하여, utmpx 파일에서 로그인 정보를 순차적으로 읽은 뒤, 현재 서버에 접속중인 사람들을 보여주는 프로그램입니다.

4.(4-4.c, 4-4) 유닉스 명령 date를 사용하면 다음과 같은 결과를 얻는다.

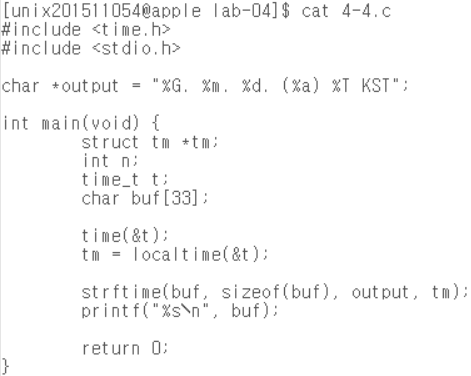


system call time()과 라이브러리 함수 localtime() 등 각종 시간관련 함수를 이용하여 date 명령과 같은 결과를 출력하는 프로그램 4-4.c를 작성하고 컴파일, 수행하여 결과를 보이시오.

출력 결과는 아래와 같고,

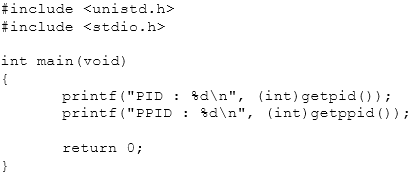


소스 코드는 아래와 같습니다.



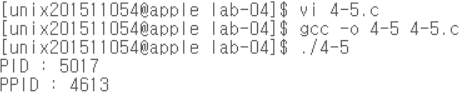
strftime을 이용하여 소스 코드를 짰고, buf는 33으로 딱 출력 결과 값 만큼 할당해주었습니다.

5.(4-5.c, 4-5) 다음과 같이 파일 4-5.c를 편집하고 4-5로 컴파일하여 수행하시오.



수행 결과를 적고, 결과를 설명하시오.

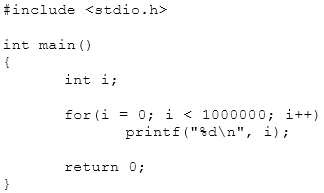
수행 결과는 아래와 같고,



PID는 현재 프로세스의 ID를 검색해서 출력해주는 것이고,

PPID는 현재 프로세스의 부모 프로세스 ID를 검색해서 출력해주는 것입니다.

6.(4-6.c, 4-6) 아래와 같이 0에서 999999까지 출력하는 프로그램을 4-6.c 로 작성하시오.

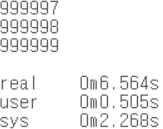


이를 컴파일하여 4-6을 만들고

$time ./4-6

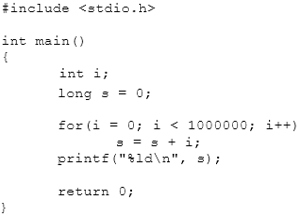
으로 수행하고 결과를 적으시오.

결과는 아래와 같고,



프로세스의 전체 시간, 사용자 모드 실행 시간, 시스템 모드 실행 시간을 출력해줍니다.

7. (4-7.c, 4-7) 아래와 같은 프로그램을 4-7.c로 작성하고

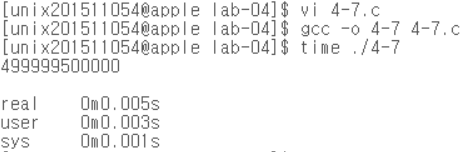


이를 컴파일하여

$time ./4-7

와 같이 수행하여 결과를 적으시오

결과는 아래와 같고,



6번과 마찬가지로 전체 시간, 사용자 모드 실행 시간, 시스템 모드 실행 시간을 출력해줍니다.

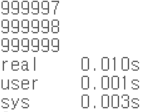
8. 위 6번과 7번의 결과를 비교하여 설명하시오.

|  |  |  |
| --- | --- | --- |
|  | 6번 | 7번 |
| Real | 0m6.564s | 0m0.005s |
| User | 0m0.505s | 0m0.003s |
| Sys | 0m2.268s | 0m0.001s |

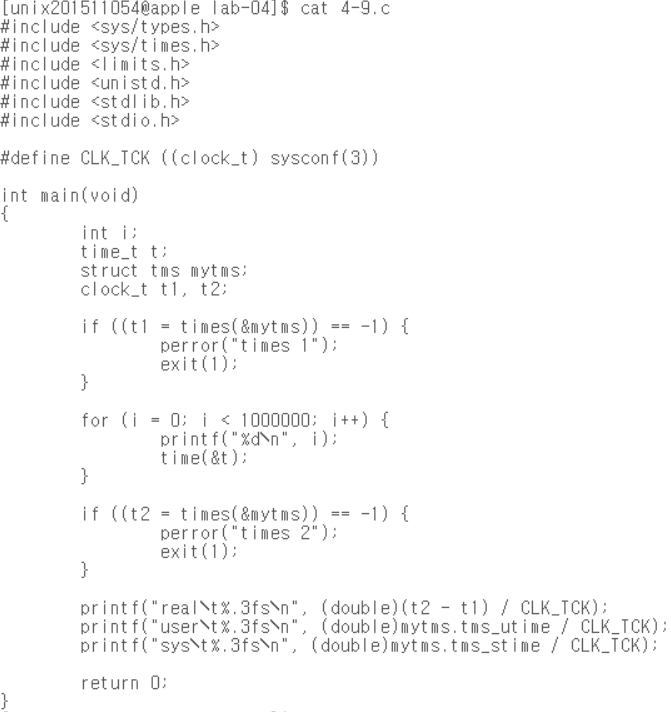
7번의 결과는 6번의 결과와는 달리 전체 시간, 사용자 모드 실행 시간, 시스템 모드 실행 시간이 모두 짧습니다. 이는 6번은 printf를 반복문으로 돌려서 실질적인 실행 시간이 오래 걸리지만, 7번은 계산 후 마지막 출력 값만 해주므로, 더 짧은 것으로 추정됩니다.

9.(4-9.c, 4-9) 프로세스의 수행시간을 측정하기 위해서는 struct tms 구조체와 times() 함수를 사용한다. 4-6.c를 수정하여 경과시간, 사용자 수행시간, 시스템 수행시간을 출력하는 프로그램 4-9.c를 작성하고 이를 컴파일, 수행하고 결과를 적으시오. 이것을 6번의 결과와 비교하여 논의하시오. (교재 5-4참조)

수행 결과는 아래와 같고,



소스 코드는 아래와 같습니다.

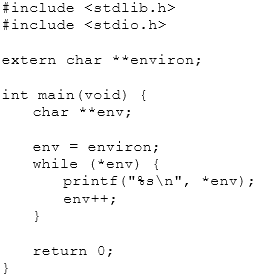


또한 이것을 6번의 결과와 비교하여 논의하자면,

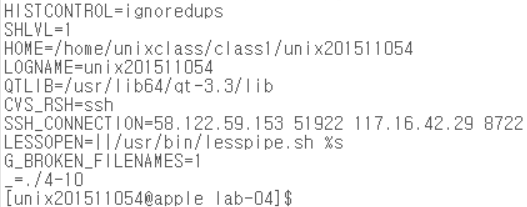
|  |  |  |
| --- | --- | --- |
|  | 6번 | 9번 |
| Real | 0m6.564s | 0.010s |
| User | 0m0.505s | 0.001s |
| Sys | 0m2.268s | 0.003s |

이번에는 6번과 9번 모두 printf를 반복문으로 돌려주지만, 9번은 6번과는 달리 프로그램 내에서 시간을 측정하는 것이라 7번과 같은 결과를 출력했다고 보아집니다.

10.(4-10.c, 4-10) 아래와 같이 프로그램 4-10.c를 작성하고 이를 컴파일 하여 수행하고 그 결과를 적으시오. (마지막 10행만)



수행 결과는 아래와 같습니다.



출력 결과로는 환경 변수 안에 있는 값들이 출력되었습니다.

끝.