시스템 프로그래밍 실습과제-2016

Lab-1 : .bashrc 파일분석

1. 현재 디렉토리의 .bachrc 파일을 분석한다.
2. bash prompt에 user-id 와 현재 디렉토리를 표시할 수 있도록 변경하시오.
- .bashrc나 .bash_profile에 PS1 환경변수를 변경한다.

* Due day : 10.7(금) 오후12:00 까지 GitHub에 업로드

Lab-2 : Directory Scan

1. 현재 디렉토리에 속한 파일명을 화면에 출력하는 프로그램(DirScan.c)을 작성한다.

- 교재 및 강의자료의 "printdir.c" 예제 프로그램을 참조할 것.
- hidden 파일 (파일명이 dot(.)로 시작하는 파일)은 제외할 것.
- 현재 디렉토리의 하부 디렉토리(sub-directory)도 재귀적으로 탐색한다. 즉, 현재 디렉토리에에 속한 디렉토리 내용도 같은 방식으로 출력한다.
- 하부 디렉토리 구분은 tab문자를 추가한다.

* Due day : 10.10(월) 오후12:00 까지 GitHub에 업로드

Lab-3 : Comparison of low-level vs. high-level file access

1. Low-level 파일함수를 사용하여 파일복사 프로그램(LowCopy.c)을 작성한다.

- 파일을 복사하는 동안 화면에 dot(.)를 일정시간 간격으로 출력하여 파일 복사중임을 표시한다.
- dot 표시 간격은 임의로 설정할 수 있다.
- 복사할 파일명은 프로그램 파라미터로 지정할 수도 있고, 프로그램 내부에 고정할 수 있다.

2. High-level 파일함수를 사용하여 파일복사 프로그램(HighCopy.c)을 작성한다.

- 파일을 복사하는 동안 low-level과 같은 방식으로 asterisk(*)를 출력한다.

3. "time" 명령어를 사용하여 실행한다.

$ time HighCopy xxx,avi // 복사대상 파일을 파라미터로 지정한 경우
$ time HighCopy // 복사대상 파일을 프로그램 내부에 고정한 경우

* Due day : 10. 14(금) 오후12:00

Lab-4 : 응용 프로그램에서 환경변수 확인 및 변경

1. getenv(), 함수를 사용하여 $HOME, $PS1, $PATH, $LD_LIBRARY_PATH 환경변수에 설정된 값을 화면에 표시한다.

2. setenv() 함수를 사용하여 $TEST_ENV 라는 환경변수를 정의하고 그 값을 1234 로 설정하고, 설정된 값을 bash에서 확인한다.

* Due day : 10.17(월)

Lab-5 : make 유틸리티 사용

1. 샘플 소스 파일 mtest.c foo.c boo.c bar.c를 만든다.

- foo.c : InFoo() 함수를 정의한다. 함수의 내용은 자유.
- boo.c : InBoo() 함수를 정의한다. 함수내용은 자유.
- bar.c : InBar() 함수를 정의한다. 함수내용은 자유.
- mtest.c : InFoo(), InBoo(), InBar() 함수를 호출한다.

2. mtest.c를 컴파일하여, mtest 라는 실행파일을 생성하는 makefile을 작성한다.
3. make 유틸리티를 사용하여 mtest를 생성하고, 쉘에서 실행한다.
4. 컴파일시 -g 옵션을 추가하여 디버그 정보를 생성한다.
5. gdb를 사용하여 mtest 파일에서 Infoo, InBoo, InBar 함수를 호출하는 과정을 trace 한다.

* Due day : 10.21(금)

Lab-6 : process monitoring

1. 아래와 같은 ptest 프로그램을 작성한다.

- ptest : 시스템의 local time 을 2초 간격으로 터미널에 표시한다.
- 기타 언급되지 않은 사항은 임의 구현 가능

2. 아래와 같은 pmon 프로그램을 작성한다. pmon 프로그램은 ptest 실행여부를 화면에 표시하고, 사용자의 명령어 입력에 따라 해당 동작을 수행한다.

- pmon : 5초간격으로 ptest 프로그램의 실행여부를 표시한다. ptest 프로세스의 존재여부를 확인하여 "running"(실행중인 경우), "not existed"(프로세스가 없는 경우) 로 표시한다.
- pmon 프로그램이 지원해야 하는 사용자 명령어. 명령어 prompt 는 ">>".

Q (quit) : pmon 프로그램 종료
K (kill) : ptest 프로그램의 강제 종료
S (start) : ptest 프로그램 실행, 이미 실행중이면, “already running” 메시지 출력.
R (restart) : 실행중인 ptest를 종료후 재실행. ptest가 없는 경우, “newly started” 메시지 출력.

- 기타 언급되지 않은 사항은 임의 구현 가능

3. pmon 프로그램2개의 프로그램을 별도 터미널에서 실행한다.

Lab-7 : Multi-processing

1. 아래와 같은 subproc 프로그램을 작성한다.

- “test_funct()” 함수를 반복적으로 호출한다.
- test_funct()함수에서는 현재 프로세스의 PID, 현재시간, 함수가 호출된 횟수를 카운트하여 출력한다.
- test_funct함수의 호출간격은 1~10초 사이의 랜덤시간으로 한다.

```
#include <time.h>
#include <stdlib.h>

srand(time(NULL));
int r = rand()%10;
sleep(r);
```

2. 아래와 같은 mproc1 프로그램을 작성한다.

- 학번과 이름을 화면에 출력한다.
- fork(), exec() 함수를 사용하여 10개의 subproc 프로세스를 실행한다.
- 각 함수 호출 카운트 값이 20이 되면 해당 프로세스를 종료시킨다.
- 모든 프로세스가 종료되면 “Finished (process)”를 출력한다.

Lab-8 : Multi-threading

1. 아래와 같은 mproc2 프로그램을 작성한다.

- 학번과 이름을 화면에 출력한다.
- pthread 라이브러리를 사용하여 Lab-7의 subproc을 10개의 thread로 실행시킨다.
- 각 함수 호출 카운트 값이 20이 되면 해당 쓰레드를 종료시킨다.
- 10개의 쓰레드가 모두 종료되면 “Finished (thread)”를 출력한다.

Lab-9 : Synchronization

1. 아래와 같은 mproc3 프로그램을 작성한다.

- 학번과 이름을 화면에 출력한다.
- 10개의 subproc을 process 또는 thread로 실행시킨다.
- 화면에 출력되는 각 함수 호출 카운트 값이 20이 되면 해당 쓰레드를 종료시킨다.
- 각 프로세스 또는 쓰레드에서 출력하는 함수 호출 카운트 값이 monotonically increased 하도록 프로세스들을 semaphore나 mutex를 사용하여 동기화시킨다. 예를 들면, 현재 출력된 카운트 값이 ‘3’ 이라면 다음에 출력되는 카운트 값은 반드시 ‘3’이상이 되어야 한다. 즉, 10개의 프로세스 또는 쓰레드가 호출되는 횟수는 항상 동일해야 한다.
- 10개의 쓰레드가 모두 종료되면 “Finished Successfully”를 출력한다.

Lab-10 : IPC-pipe

1. 아래와 같은 ipc_producer, ipc_consumer 프로그램을 작성한다.

- ipc_producer 프로그램은 ‘start’라는 키입력을 받으면 자신의 학번과 PID를 ipc_consumer에 보낸다.
- ipc_consumer 프로그램은 학번을 받으면 화면에 학번과 ipc_producer의 PID를 화면에 출력하고, 자신의 이름과 자신의 PID를 ipc_producer에 보낸다.
- 자신의 이름과 ipc_consumer의 PID를 받은 ipc_producer는 화면에 자신의 PID와 ipc_consumer의 PID, 이름과 학번을 화면에 출력한다.

2. ipc_producer와 ipc_consumer의 IPC 방법을 pipe를 사용하여 구현한다.

Lab-11 : IPC-shared memory

1. Lab-10의 ipc_producer, ipc_consumer 프로그램이 사용하는 IPC 수단으로 shared memory를 사용하여 구현하여 ipc_producer2, ipc_consumer2를 작성한다.

Lab-12 : IPC-message queue

1. Lab-10의 ipc_producer, ipc_consumer 프로그램이 사용하는 IPC 수단으로 message queue를 사용하여 구현하여 ipc_producer3, ipc_consumer3을 작성한다.

Lab-13 : IPC-socket

1. Lab-10의 ipc_producer, ipc_consumer 프로그램이 사용하는 IPC 수단으로 socket을

사용하여 구현하여 ipc_producer4, ipc_consumer4를 작성한다

Lab-14 : ELF 파일분석

1. ELF 파일을 분석하여 ELF header, Program header table, Section header table 내용을 화면에 표시한다. 프로그램명은 elfreader, 분석할 ELF파일을 파라미터로 주어지며, 옵션은 e(ELF header), p(program header table), s(section header table)이며, 동시에 사용가능하다.

$ elfreader xxx.elf // ELF header만 표시
$ elfreader -e xxx.elf // ELF header만 표시
$ elfreader -ep xxx.elf // ELF header + program header table 표시
$ elfreader -eps xxx.elf // ELF header + program header table + section header table 표시

기타 다른 옵션 조합도 가능.