**Deadlock Detection**

**결과 보고서**

김세란

2018310520

**목차**

**1. 설계/구현 내용 설명**

**1) 개발 환경**

**2) 설계&구현: 데이터,알고리즘**

**2. 다양한 입력에 대한 실행결과**

**3. 제출물 내 각 파일에 대한 설명 (제출물 내 소스코드는 오직 dd.c 파일 하나로 구성됨.)**

**1. 설계/구현 내용 설명**

**1) 개발 환경**

운영체제: Linux 16.04 LTS

소스코드편집기:VSCode

프로그래밍언어: C언어

컴파일러: GCC

**2)설계&구현**

dd.c의 코드를 보면서, 해당 개발물의 설계와 그에 따른 구현을 설명한다.

|  |  |
| --- | --- |
| **1. 헤더파일** | |
|  | 입출력을 위한 stdio.h 와 malloc을 위한 stdlib.h |
| **2. 데이터** | |
|  | 옆의 데이터를  Input.txt 파일로 부터  입력 받는다.  입력 받은 정보로 해당 데이터에 대해 동적 할당을 진행하였다.  입력시 invalid input file에 대해 예외처리를 하였다. |
| **3. 알고리즘: graph reduction** | |
| Graph reduction을 진행하는 과정에서 process의 상태를 세가지 상태로 구분한다.  'b'(blocked), 'u'(unblocked), 'd'(done)의 세가지 상태로 나눈다.  Grepah Reuction 알고리즘은 deadlock/non deadlock이 결정될 때까지 다음 과정을 반복한다.  **1.** resource별 available한 unit 개수 구하기.  **2.** done이 아닌 프로세스에 대해 새롭게 unblocked/blocked 를 구분하기.  **3-1.** unblocked 상태의 프로세스가 여러 개인 경우 하나로 바꾸어주기.  **3-2.** deadlock detection  4. deadlock/non deadlock이 탐지되지 않았다면, reducing 진행.  **아래에 4 단계를 좀 더 구체적으로 설명한다.** | |

**4단계:**

|  |  |
| --- | --- |
| **1. resource별 available한 unit 개수 구하기.** | |
| **텍스트이(가) 표시된 사진  자동 생성된 설명** | 매 루프의 첫 단계에서  자원 별로  할당가능한 자원의 유닛 개수를 구해준다.  할당 가능한 자원 k의 유닛 개수  = 자원 k 의 유닛 개수 – 모든 프로세스에 대해 자원 k가 할당한 유닛 개수 |
| **2. done이 아닌 프로세스에 대해 새롭게 unblocked/blocked를 구분한다.** | |
| **텍스트이(가) 표시된 사진  자동 생성된 설명** | Done 상태가 아닌 프로세스에 대해,  정의에 따라 존재하는 모든 자원에 대해 해당 프로세스가 요청한 자원의 유닛 개수가 그 자원에서 할당 가능한 유닛 개수보다 크면 unblocked 상태, 그렇지 않으면 blocked 상태이다. |
| **3-1. unblocked가 여러 개인 경우** | |
|  | 이후 deadlock detection 단계에서 unblocked 상태의 프로세스가 두개 이상인 경우 혼란이 생길 수 있다. 따라서 unblocked 상태의 프로세스가 두 개 이상인 경우, 우선 하나로 바꾸어준다.  어짜피 blocked 상태로 바꾸어 준다고 해서 reduction이 끝날 때까지 unblocked 상태로 변하지 않고, 이후 단계에서 다시 Unblocked -> done의 상태로 변하기 때문에 상관이 없다. |
| **3-2. deadlock detection:** | |
| **텍스트이(가) 표시된 사진  자동 생성된 설명** | 모든 경우의 수는 다음과 같이 분류할 수 있다.  1. 모두 d 인 경우  2. 모두 b 인 경우  3. d 또는 b 로만 이루어진 경우  4. u가 존재하는 경우 ((u,b,d)로 이루어짐, (u,d) 또는 (u,b)로 이뤄짐)  이중 첫 번째의 경우 graph reduction 결과 completely reducible하며, non-deadlock상태라는 것을 의미한다. 두번째와 세번째의 경우는 irreducible한 결과로 deadlock 상태라는 것을 의미한다.  마지막 u가 존재하는 경우는 아직 결론이 나지 않았으므로 4번 graph reduction 진행 후 다시 1번부터 반복해야한다. |
| **4. reducing:** | |
| 텍스트이(가) 표시된 사진  자동 생성된 설명 | 하나 선정된 unblocked 상태의 프로세스에 할당/요청된 자원을 지우고 그 프로세스의 상태를 done으로 만든다. |

**2. 다양한 입력에 대한 실행 결과**

**Unblocked인 process를 하나 씩 선택하면서, 루프를 돌 때마다 각 프로세스의 상태를 표시하였다. 마지막에는 deadlock/non-deadlock 상태임과 deadlock상태일때 deadlocked 프로세스를 밝혔다.**

|  |
| --- |
| **입력1:**  **3 3 3 2 2**  **2 1 0**  **1 0 0**  **0 1 1**  **1 1 0**  **0 2 1**  **0 0 1** |
| **실행결과:** |
| **입력2:**  **5 3 7 2 6**  **0 1 0**  **2 0 0**  **3 0 3**  **2 1 1**  **0 0 2**  **0 0 0**  **2 0 2**  **0 0 0**  **1 0 0**  **0 0 2** |
| **실행결과:** |
| **입력 3:**  **5 3 2 2 1**  **0 1 0**  **0 0 0**  **0 0 1**  **1 0 0**  **0 1 0**  **1 0 0**  **0 1 0**  **0 2 0**  **0 1 0**  **0 0 1** |
|  |
| **입력 4:**  **4 3 2 1 2**  **1 0 0**  **0 1 1**  **1 0 0**  **0 0 0**  **0 1 0**  **2 0 1**  **0 0 1**  **1 0 0** |
|  |

**3. 제출물 내 각 파일에 대한 설명**

제출물 내 dd.c의 C파일 하나로 구성되어있다.

컴파일러를 이용하여 gcc dd.c -o dd 로 dd 실행파일을 생성시킨 후 ./dd 명령어로 실행 파일을 실행할 수 있다.

dd.c 에서 읽어들이는 파일은 텍스트 파일 형태이며, Input.txt 이란 파일명을 갖는다.