운영체제

Dining Philosopher

학번 : 20143104
이름 : 조승현

Lock lock1, lock2, lock3 : 젓가락을 집는 것을 mutual exclusive하게 만드는 lock
CondVar c1, c2, c3 : 젓가락을 기다리는 condition variable
r1, r2, r3 : 프로세스들끼리 공유하기 위해 만든 젓가락 파일

< Deadlock을 구현한 코드 >

```
void take_r1(){
  pid_t pid;
  pid = getpid();
  Acquire(&lock1);
  while(Load("r1")==0){  // r1을 누가 가져갔음을 의미한다.
    printf("%d 가 R1을 기다림\n", pid);
    Wait(&c1, &lock1);
    printf("%d 가 R1을 기다리다가 깨어남\n", pid);
  }
  Store("r1", 0); //r1의 take 상태를 나타낸다.
  printf("%d가 R1을 가져옴\n", pid);
  Release(&lock1);
}
```

```
void put_r1(){
  pid_t pid;
  pid = getpid();
  Acquire(&lock1);
  Store("r1", 1); //r1의 put 상태를 나타낸다.
  Signal(&c1);  // r1을 기다리는 누군가가 있다면 signal
  printf("%d 가 R1을 놓음\n", pid);
  Release(&lock1);
}
```

```
void phil_a(){
  take_r1();
  printf("%d가 생각을 시작함\n", pid);
  sleep(1);
  printf("%d가 생각을 멈춤\n", pid);
  take_r2();
  printf("%d가 먹기 시작\n", pid);
  sleep(1);
  printf("%d가 먹기를 멈춤\n", pid);
  put_r1();
  put_r2();
}
```

< Prevention을 구현한 코드 >

take 함수와 put 함수는 deadlock이 걸리는 코드와 동일 하지만 A,B,C 각각 젓가락을 잡는 일정한 순서를 부여한다.(숫자가 작은 젓가락을 먼저 잡도록)
A: 1->2 B: 2->3 C: 1->3

```
void phil_a(){
   take_r1();
   printf("%d가 생각을 시작함\n", pid);
   sleep(1);
   printf("%d가 생각을 멈춤\n", pid);
   take_r2();
   printf("%d가 먹기 시작\n", pid);
   sleep(1);
   printf("%d가 먹기를 멈춤\n", pid);
   put_r1();
   put_r2();
}
```

```
void phil_b(){
   take_r2();
   printf("%d가 생각을 시작함\n", pid);
   sleep(1);
   printf("%d가 생각을 멈춤\n", pid);
   take_r3();
   printf("%d가 먹기 시작함\n", pid);
   sleep(1);
   printf("%d가 먹기를 멈춤\n", pid);
   put_r2();
   put_r3();
}
```

```
void phil_c(){ // c가 1번을 잡은 후 3번을 잡게 구현했다.
  pid_t pid;
  pid = getpid();
  take_r1();
  printf("%d가 생각을 시작함\n", pid);
  sleep(1);
  printf("%d가 생각을 멈춤\n", pid);
  take_r3();
  printf("%d가 먹기 시작함\n", pid);
  sleep(1);
  printf("%d가 먹기를 멈춤\n", pid);
  put_r1();
  put_r3();
}
```

< Avoiding을 구현한 코드 >

젓가락 집는 순서 : Deadlock 상황과 동일
Lock lock_all : Avoiding 상황시에 mutual exclusive하기 위해 만든 변수
Condvar c_all : Avoiding 상황시에 c1,c2,c3와 구별하기 위해 따로 만든 변수
resc : 남아있는 젓가락의 개수를 나타냄(프로세스끼리 공유하기 위해 파일로 구현함)
int num : 현재 Philosopher가 가지고 있는 젓가락 개수를 나타내는 변수

```
void take_r1(){
  Acquire(&lock1);
  Acquire(&lock_all);
  while( (Load("resc") == 1 && num == 0) || (Load("r1") == 0) ){ // Avoiding 상황이거나 젓가락이 없는 상황이라면
    printf("%d 가 R1을 기다림\n", pid);
    Release(&lock1); Release(&lock_all);
    if(Load("r1") == 0) Wait(&c1, &lock1); //젓가락이 없기때문에 젓가락의 condvar에서 기다리게 된다.
    else Wait(&c_all, &lock_all); // avoiding 상황의 condvar에서 기다리게 된다.
    Acquire(&lock1); Acquire(&lock_all);
    printf("%d 가 R1을 기다리다가 깨어남\n", pid);
  }
  Store("r1", 0);
  num+=1; // 자신이 가지고 있는 개수를 센다.
  sub("resc", 1); // 남아있는 전체 젓가락의 개수에서 -1
  printf("%d 가 R1을 가져감\n", pid);
  Release(&lock_all);
  Release(&lock1);
}
```

```
void put_r1(){
  Acquire(&lock1);
  Acquire(&lock_all);
  Store("r1", 1); //현재 젓가락의 상태
  num-=1; //놓았으므로 가지고 있는 개수 -1
  add("resc", 1); // 전체 남아있는 개수 +1
  Signal(&c1);
  Signal(&c_all); // avoiding 상황으로 기다리고 있는 사람도 깨워준다
  printf("%d 가 R1을 놓음\n", pid);
  Release(&lock_all);
  Release(&lock1);
}
```