```
#define NUM_PHIL 5
int forks[NUM_PHIL] = -1;

//Initialise all philosophers status' to false
bool pthinking[NUM_PHIL] = false;
bool phungry[NUM_PHIL] = false;
bool peating[NUM_PHIL] = false;



init{
        atomic{
                int i = 0;

                //Create each philosopher, run P
                do
                :: i < NUM_PHIL ->
                        run P(i);
                        i++;

                :: else ->
                        break;
                od;
        }
}

proctype P(int i){

        //Right fork at index i
        int right = i;
        //Left fork at (i+1)%N
        int left = (i+1)%NUM_PHIL;

think:
        atomic{
                peating[i] = false;
                pthinking[i] = true;
        };

hungry:
        atomic{
                pthinking[i] = false;
                phungry[i] = true;
        };

        //Un-deterministically decide whether to check left/right first
        if
        :: skip;
                //Attempt to pickup left fork
                atomic { forks[left] == -1 -> forks[left] = i};
                //Attempt to pickup right fork
                atomic { forks[right] == -1 -> forks[right] = i};

        :: skip;
                //Attempt to pickup right fork
                atomic { forks[right] == -1 -> forks[right] = i};
                //Attempt to pickup left fork
                atomic { forks[left] == -1 -> forks[left] = i};
        fi;

eating:
        atomic {
                phungry[i] = false;
                peating[i] = true;
        };

done:
        forks[right] = -1;
        forks[left] = -1;
        goto think;
}
```