The Dining Savages Problem

written by Nik Tsonev

Intro

you have n amount of savages at a table, each are eating from the same pot of food. When the food runs out, one of the savages must notify the cook to refillthe pot.

Solution

note that in this case, there is one main thread (cook) and many other threads (savages). For this task we would need one mutex and two condition variables empty, and full

so the cook locks and waits until the food is 0 and the cond var is_empty that is signaled by one of the savages. he then fills up the pot and broadcast to all of them that its full using the cond var is_full.

each savage locks, waits until the pot is not empty using the cond var, then eats and each one checks if there is no food left, if there is no food left they signal using the is_empty cond var

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>

#define POT_SIZE 3

pthread_mutex_t m;
pthread_cond_t is_empty;
pthread_cond_t is_full;

int servings = 0;
```

```
void* cook(void *arg) {
        (void) arg;
        while (1) {
                pthread_mutex_lock(&m);

                while (servings != 0)
                        pthread_cond_wait(&is_empty, &m);

                servings = POT_SIZE;

                sleep(1); // time taken to refil
                pthread_cond_broadcast(&is_full);
                pthread_mutex_unlock(&m);
        }

        return NULL;
}

void* savage(void *arg) {
        int id = *(int*) arg;
        while (1) {
                pthread_mutex_lock(&m);

                while (servings == 0)
                        pthread_cond_wait(&is_full, &m);

                servings--;
                printf("%d is eating\n");
                sleep(1); // time taken to eat
                if (servings == 0)
                        pthread_cond_signal(&is_empty);
                pthread_mutex_unlock(&m);
        }
        return NULL;
}
```