The forest problem: mutex, cond variable

#include <stdio.h>

#include <pthread.h>

#define BEES 100

#define BEARS 5

#define MEAL 20

int honey = 300;

pthread\_mutex\_t m;

pthread\_cond\_t c;

void\* bee(void\* a) {

while(1) {

pthread\_mutex\_lock(&m);

honey++;

printf("+");

pthread\_mutex\_unlock(&m);

}

return NULL;

}

void\* bear(void\* a) {

while(1) {

pthread\_mutex\_lock(&m);

if(honey >= MEAL)

honey -= MEAL;

else {

pthread\_cond\_signal(&c);

printf("!");

}

pthread\_mutex\_unlock(&m);

}

return NULL;

}

void\* ranger(void\* a) {

while(1) {

pthread\_mutex\_lock(&m);

while(honey >= MEAL)

pthread\_cond\_wait(&c, &m);

honey += 10 \* MEAL;

printf(" ");

pthread\_mutex\_unlock(&m);

}

return NULL;

}

int main() {

pthread\_t bees[BEES], bears[BEARS], rngr;

int i;

pthread\_mutex\_init(&m, NULL);

pthread\_cond\_init(&c, NULL);

/\*creating the ranger thread\*/

pthread\_create(&rngr, NULL, ranger, NULL);

/\*creating the bee threads\*/

for(i = 0; i < BEES; i++) {

pthread\_create(&bees[i], NULL, bee, NULL);

}

/\*creating the bear threads\*/

for(i = 0; i < BEARS; i++) {

pthread\_create(&bears[i], NULL, bear, NULL);

}

/\*joining the ranger thread\*/

pthread\_join(rngr, NULL);

/\*joining the bee threads\*/

for(i = 0; i < BEES; i++) {

pthread\_join(bees[i], NULL);

}

/\*joining the bear threads\*/

for(i = 0; i < BEARS; i++) {

pthread\_join(bears[i], NULL);

}

pthread\_cond\_destroy(&c);

pthread\_mutex\_destroy(&m);

return 0;

}