

## Processes and Threads

1

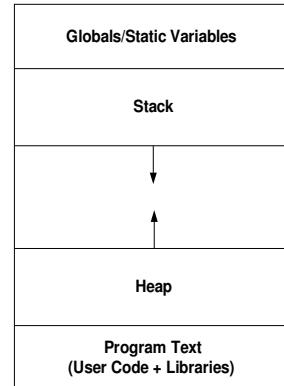


### Process

- ‘Heavy-weight’ unit of computation
- Process descriptor
  - Object program (Program text)
  - Data segment
  - Stack
  - Heap
  - Process Status Word (PSW) – executing, waiting, ready
  - Resources acquired

## Process contents

- Memory for each process contains
  - Program text
  - Globals/static variables
  - Stack
  - Heap

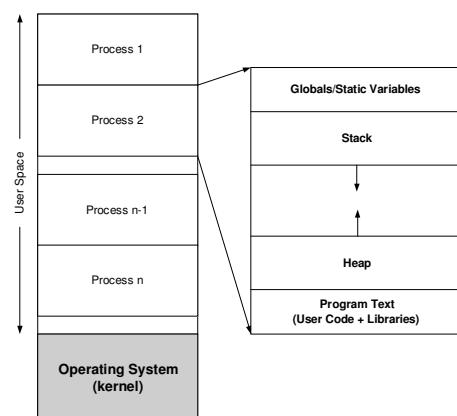


September 1, 2002

CS 3204: Operating Systems, Fall 2002  
© Mir Farooq Ali, 2002

3

## Main Memory



September 1, 2002

CS 3204: Operating Systems, Fall 2002  
© Mir Farooq Ali, 2002

4

## Process Control Block (PCB)

- Also called Process Descriptor
- Each process has per-process state maintained by the OS
  - Identification: process, parent, user, group, etc.
  - Address space: virtual memory, memory limits
  - I/O state: file handles (file system), communication endpoints (network), etc.
  - Accounting information
  - Program counter, Stack counter
- Details in later chapter

September 1, 2002

CS 3204: Operating Systems, Fall 2002  
© Mir Farooq Ali, 2002

5

## Thread

- Thread: light-weight process
  - OS maintains minimal internal state information
- Usually instantiated from a process
- Each thread has its OWN unique descriptor
  - Data, Thread Status Word (TSW)
- SHARES with the parent process (and other threads)
  - Program text
  - Resources
  - Parent process data segment

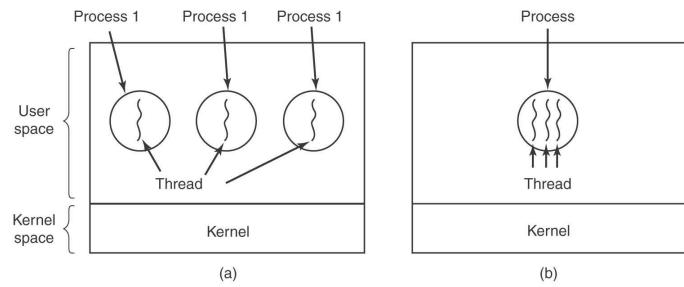
September 1, 2002

CS 3204: Operating Systems, Fall 2002  
© Mir Farooq Ali, 2002

6

## Process Vs Threads

- Processes require substantially more OS overhead in creation and maintenance



Taken from Modern Operating Systems, 2<sup>nd</sup> Ed, Tanenbaum, 2001

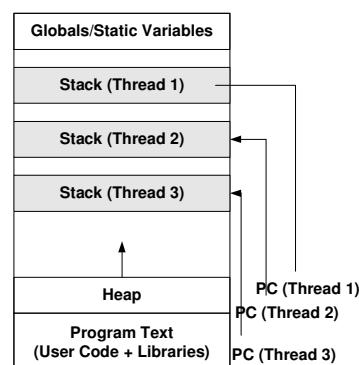
CS 3204: Operating Systems, Fall 2002  
September 1, 2002

© Mir Farooq Ali, 2002

7

## Thread space

- Data is shared among all threads
- Each thread maintains its own stack
- Each thread has its own Program Counter (PC)



CS 3204: Operating Systems, Fall 2002  
September 1, 2002

© Mir Farooq Ali, 2002

8

## wait()

- Used by parent process to wait on ONE child process to finish
- `int wait(&status);`
- return value of wait is the process id of child process that just finished
- if no child processes, wait returns –1 immediately

September 1, 2002

CS 3204: Operating Systems, Fall 2002  
© Mir Farooq Ali, 2002

9

## wait() ... ctd

- wait returns value if child process
  - called function exit()... or terminated normally
  - gets terminated by a signal
- returns exit status of child in variable `status`

September 1, 2002

CS 3204: Operating Systems, Fall 2002  
© Mir Farooq Ali, 2002

10

## waitpid()

- Used by parent to wait on a specific child process to terminate indicated by pid
- `int waitpid(pid, &status, options)`
- `pid`: process id of the child process parent waits on

September 1, 2002

CS 3204: Operating Systems, Fall 2002  
© Mir Farooq Ali, 2002

11

## pipes

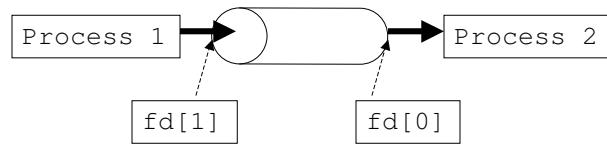
- One form of inter-process communication (IPC)
- follows message-passing paradigm of IPC

September 1, 2002

CS 3204: Operating Systems, Fall 2002  
© Mir Farooq Ali, 2002

12

## pipes...ctd



```
int fds[2];
retval = pipe(fd);
```

- creates two file descriptors, one for reading, the second for writing

September 1, 2002

CS 3204: Operating Systems, Fall 2002  
© Mir Farooq Ali, 2002

13

## pipes...ctd

```
int fds[2]; char s[100];
retval = pipe(fds);
pid = fork();
if(pid != 0){ /* parent process */
    write(fds[1], "hello", 6);
}
else { /* child process */
    read(fds[0], s, 100);
    printf("Read %s\n", s);
}
```

September 1, 2002

CS 3204: Operating Systems, Fall 2002  
© Mir Farooq Ali, 2002

14



## What are pthreads?

- A standardized programming interface
- For UNIX systems, specified by the IEEE POSIX 1003.1c standard (1995).
- Implementations which adhere to this standard are referred to as POSIX threads, or Pthreads.

September 1, 2002

CS 3204: Operating Systems, Fall 2002  
© Mir Farooq Ali, 2002

15



## Why pthreads over fork()?

- Primary reason is performance gains
- Less OS overhead in creating a new thread
- All threads use same address space, so communication between threads is easier
- `$gcc -o firstthread firstthread.c -lpthread`

September 1, 2002

CS 3204: Operating Systems, Fall 2002  
© Mir Farooq Ali, 2002

16

## pthread creation

- Use `pthread_create` function

```
pthread_create(thread, attr, routine, arg)
```

- `thread`: Name of this thread
- `attr`: Thread attributes
- `routine`: function that gets executed once thread is started
- `arg`: A single argument to be passed to routine, cast as pointer of type `void`, passed by reference.
  - For multiple arguments, bundle them up in a struct and pass struct to routine

September 1, 2002

CS 3204: Operating Systems, Fall 2002  
© Mir Farooq Ali, 2002

17

## First pthread program

```
#include <pthread.h>
#include <stdio.h>
#define NUM_THREADS      5
int main()
{
    pthread_t threads[NUM_THREADS];
    int rc, t;
    for(t=0;t < NUM_THREADS;t++){
        printf("Creating thread %d\n", t);
        rc = pthread_create(&threads[t], NULL, PrintHello,
                           (void *)t);
    }
    pthread_exit(NULL);
}
```

September 1, 2002

CS 3204: Operating Systems, Fall 2002  
© Mir Farooq Ali, 2002

18

## First pthread program... ctd

```
void *PrintHello(void *threadid)
{
    printf("%d: Hello World!\n", threadid);
    pthread_exit(NULL);
}
■ pthread_exit(void *status) : Used to explicitly
  terminate a thread
■ Thread can use the status variable to specify its status; pass
  data to 'joining' threads
```

September 1, 2002

CS 3204: Operating Systems, Fall 2002  
© Mir Farooq Ali, 2002

19

## pthreads... ctd

- `pthread_join()` : Analogous to `wait()` for processes.
- Allows threads to 'join' to form single thread of execution

September 1, 2002

CS 3204: Operating Systems, Fall 2002  
© Mir Farooq Ali, 2002

20

## Second example

```
#include <pthread.h>
#include <stdio.h>

int main(void) {
    int N = 8;
    pthread_t hThread;    int fact;
    pthread_create(&hThread, NULL, (void *)ChildThread,
                  (void *)N);
    pthread_join(hThread, (void *)&fact);

    printf("Factorial of N = %d\n", fact); return 0;
}
```

September 1, 2002

CS 3204: Operating Systems, Fall 2002  
© Mir Farooq Ali, 2002

21

## Second example... ctd

```
void ChildThread(int N) {
    int i;    int fact = 1;

    for(i=1;i<=N;++i)
        {    fact*=i;    }

    pthread_exit((void *)fact);
}
```

September 1, 2002

CS 3204: Operating Systems, Fall 2002  
© Mir Farooq Ali, 2002

22

## Reference for pthreads

- Posix threads programming

<http://www.llnl.gov/computing/tutorials/workshops/workshop/pthreads/MAIN.html#Pthread>

- Introduction to pthreads

<http://phoenix.liunet.edu/~mdevi/pthread/Intro.htm>