Array based scheduling

xv6 uses a simple array to manage the processes, and the scheduler iterates through this array to find the next runnable process.

The maximum number of processes support in xv6 defined in param.h is as below

```
#define NPROC       64  // maximum number of processes
```

This macro is used to create ptable (process table) is as below

```
struct {
  struct spinlock lock;
  struct proc proc[NPROC];
} ptable;
```

Process State Definition

In proc.h, the process structure is defined, including the state of the process:

```
struct proc {
  // ...
  enum procstate state; // Process state
  // ...
};
```

Scheduler

The scheduler in proc.c iterates through the array of processes, looking for a runnable process:

```
for(p = ptable.proc; p < &ptable.proc[NPROC]; p++){
  if(p->state != RUNNABLE)
    continue;
  // ...
}
```

Not just the scheduler but other functions like exit(), kill() and wait() uses this macro NPROC

Adding/Removing a Process from Runnable State: When a process is created, exits, or goes to sleep, its state is updated in proc.c.

So, in conclusion, the management of runnable processes in xv6 is handled mainly through the proc.c file, with the process structure defined in proc.h.

Traditional doubly-linked list kind of implemenation is not present, but rather a simple and efficient array-based structure.